(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 October 2001 (11.10.2001)

PCT

(10) International Publication Number
WO 01/76119 A2

(51) International Patent Classification[7]: H04L

(21) International Application Number: PCT/US01/10723

(22) International Filing Date: 2 April 2001 (02.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/194,254 3 April 2000 (03.04.2000) US
09/661,882 14 September 2000 (14.09.2000) US

(71) Applicants and
(72) Inventors: **STARK, Juergen** [US/US]; 930 Vernon Avenue, Glencoe, IL 60022 (US). **GOREN, Craig** [US/US]; 1613 N. Sedgwick, Chicago, Il 60614 (US).

(74) Agent: **PINE, Jeffrey, A.**; Baniak Pine & Gannon, 150 N. Wacker Drive, Suite 1200, Chicago, IL 60606 (US).

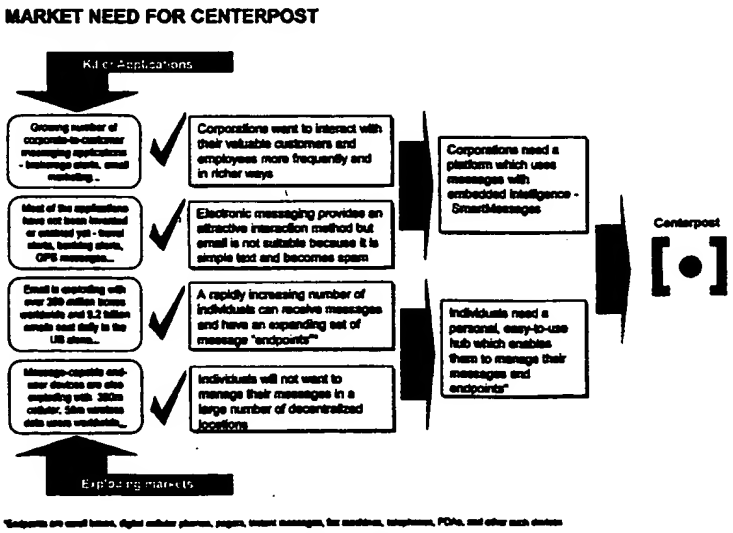
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND SYSTEM FOR CONTENT DRIVEN ELECTRONIC MESSAGING



(57) Abstract: A method and system of electronic messaging is disclosed in which the electronic messaging system is configured such that the message content contains information which will drive the routing decision process. The message itself can be prioritized and routed to a pre-defined communications device or application, even before the recipient of the message has read the message. The sender focuses on the content and MessageML provides the constructs to tag the content and properly transform it into a viewable format for delivery to the target communications device. A method is disclosed for providing content driven electronic messaging that enables individuals to receive XML electronic messages using an electronic messaging system. Once an Informant Stylesheet and SmartMessage Stylesheet are created, a SmartMessage can be sent to a MessageML Service Provider, who will receive the SmartMessage, process it, and deliver the SmartMessage to a user's defined endpoint.

WO 01/76119 A2

METHOD AND SYSTEM FOR CONTENT DRIVEN ELECTRONIC MESSAGING

This non-provisional application is based on the provisional patent application Serial No. 60/194,254, entitled Consumer XML Message Processing Platform, filed on April 3, 2000.

Cross reference is made to a related invention disclosed in U.S. patent application entitled Individual XML Message Processing Platform, filed concurrently, the subject matter of which is owned by the present applicants and the teachings of which are incorporated herein by reference.

Cross reference is also made to a related invention disclosed in U.S. patent application entitled Method of Controlling Access to Personal Resources, filed concurrently, the subject matter of which is owned by the present applicants and the teachings of which are incorporated herein by reference.

Background of the Invention

The present invention pertains to a system of electronic messaging, and more specifically the present invention pertains to a system and method of electronic messaging configured such that the message contains XML tags which will drive how the message is processed, including how it is routed, formatted, displayed, organized, and otherwise handled. The system, MessageML, operates in conjunction with a specialized message processing platform to enable individuals to receive, store, synthesize and intelligently process XML-based electronic messages and standard SMTP electronic mail (email) messages from corporations or any other senders in a way that fully integrates individuals' existing email boxes, cell phones, pagers, fax machines, telephones, and other devices. MessageML provides the constructs to tag the message content so that the

processing platform understands the meaning of the content and thus how the message should be processed. The message processing platform is described in a patent application filed concurrently herewith, entitled Individual XML Message Processing Platform, and incorporated herein by reference.

Despite the limitations of existing technologies, the electronic communications market is enormous and growing very rapidly. There are over 260 million email boxes worldwide and 9.2 billion messages sent daily in the United States alone. Mobile data appliances are likewise growing rapidly with over 208 million digital wireless phone subscribers worldwide today. Similarly, Internet appliances are expected to explode to 55 million connected devices by 2002. As shown in figure 1, there exists a need for a new communications platform capable of fully integrating individuals' existing email boxes, cell phones, pagers, fax machines, telephones, and other devices, to take full advantage of these exploding markets. In order to do this intelligently, a more robust, forward-looking messaging structure is needed.

A novel electronic messaging system is needed by businesses and consumers due to 1) inherent limitations of current electronic mail systems, 2) lack of integration between consumer "endpoints" (email addresses, wireless devices, fax machines, etc.) and 3) exploding electronic communications complexity.

First, email is not a suitable medium for corporations to interact with their customers in anything but a "newsletter" fashion. Email does not, by definition, contain anything more intelligent than a sender's address, an urgent/non-urgent tag, a subject line, and a text body. Email, by definition, typically goes into a user's single inbox where it is opened, read, and stored or deleted. Once more and more corporations send their customers email messages, the receiver quickly has a problem with inbox overload. Email also cannot be relied upon for urgent messages or alerts since users check email with varying frequency. Further, email cannot be easily automatically sorted, synthesized, filed, re-formatted or summarized.

Today, in order to automatically file or process incoming email, a user has to manually set up message-specific rules such as a search of the subject line or specify an action based on a specific sender's address. As corporations and individuals send

increasing volumes of electronic messages, a more intelligent way to process, store, and synthesize these messages is needed. It will not be convenient, for example, to open every message from a person's credit card vendor to see that payments have been posted, transactions have been processed, or bills have been mailed. These messages should be intelligently stored so that the consumer can view a synthesized status or see detail if desired.

Second, many individuals use a variety of electronic communication addresses for their email boxes, wireless phones, pagers, fax machines, instant messaging, etc. There is no way to effectively and intelligently integrate those devices: pagers, email boxes, telephones, and fax machines are ignorant of each other. There are webmail and email solutions that will forward email messages to pagers or cell phones or convert email text to a fax message or a voice message. None of these solutions intelligently determine the correct destination of each specific incoming message without the user's manual control or message-specific rules being setup beforehand. Existing services, for example, will not automatically understand that a flight cancellation alert should go to any device a receiver has that is likely to deliver the message immediately but a special fares notice from the same airline should go to a lower immediacy, less intrusive device. Many unified messaging solutions force users to give up existing addresses and phone numbers and do not process intelligent incoming messages. These platforms only enable the receiver to convert and listen to messages from a variety of platforms. While some will find a subscriber by trying multiple phone numbers, there is no real intelligent routing based on the content of the incoming message since the incoming message is generally a voicemail or an email.

Third, communications complexity is exploding for corporations and individuals. New "connected" devices are being introduced every day – each with different capabilities, formatting, protocol requirements, and addresses. Individuals have new choices in the types of information they can receive and the devices upon which that information can be received and viewed. Existing message connections between individuals and/or corporations are typically point-to-point. If a corporation wants to send a message to a customer's pager, that corporation sends a message directly to that

pager's address. Likewise if an individual wants to send a text message to a friend's specific endpoint, that person in many cases has to know and remember multiple email addresses, a PCS wireless phone text address, a fax number, a pager number, etc. If the receiving individual ever changes pagers or wireless phone providers, that individual has to remember to provide all possible sending parties with the new device-specific address.

As wireless PDAs, cars, home appliances, and other devices all begin to have their own electronic messaging address, the existing point-to-point, address-specific messaging approach will become extremely burdensome and complex for senders and receivers. Additional complexity will be generated by the introduction of features such as device sensing and endpoint location sensing technologies. This problem is exacerbated by the need to limit access to a specific individual's communications world but enable access for friends and family without having to share passwords. Figure 2 shows these above-listed limitations on the current state of electronic messaging.

Certain of the advanced messaging capabilities enabled by MessageML can be performed today within corporations. For example, intra-company messaging on a single platform often provides for sophisticated formatting, integrated calendar management, and other applications. However, these capabilities are lost entirely once messages are sent to recipients outside the corporate network. To address this problem, a new messaging standard is needed. MessageML will not only give all users the capabilities that currently exist on intra-company messaging platforms, but also add dramatically to that level of functionality.

Summary of the Invention

The present invention is a novel system and method for transmitting electronic messages. The MessageML system is based upon the eXtensible Markup Language (XML). The system is a precise XML vocabulary that can be utilized by a messaging platform to processes messages based on the meaning of their content and interpretation of the accompanying messaging vocabulary. The system provides a configuration whereby the message content contains information to, among other things, drive the how the message is processed. The sender of the message creates the actual content of the

message and uses the MessageML vocabulary to tag the content with meaning so that the message may be properly processed.

MessageML is a new electronic communications standard that applies XML, the widely accepted language of e-commerce, to messaging. MessageML is a precise XML vocabulary applied to messages to individuals. Unlike emails based on the current SMTP standard, which simply contain text content, MessageML messages contain a set of self-descriptive attributes that convey what the message's content means. Processing engines can use these attributes to intelligently manipulate and process the message, permitting advanced functionality such as routing of messages to any electronic device or messaging account; automatic formatting for various communications devices; automatic self-organization of messages within inboxes; and numerous other applications.

MessageML messages contain embedded XML tags that describe certain attributes of that message. A MessageML message processing platform matches the attributes of an incoming MessageML message with information about a recipient's communication devices and how they are used to intelligently route, format, and deliver the message. More specifically, messages composed using the MessageML vocabulary and architecture contain raw data and XML tags that describe that message's particular attributes, such as its reach, immediacy, sensitivity, content, expiration, and context. The message processor examines this meta-data in processing the message. The message processor also interrogates meta-data about the sender and about the type of message, which are contained in stylesheets residing on the sender's server. The sender attributes comprise items such as the sender's name, website address, and industry category, while the message type attributes include the type of activity the message represents (e.g. Travel Itinerary) and how those messages should be displayed on various communications devices. By applying the sender's stylesheets to the smart message itself and factoring in the recipient's device preferences, the message processor determines how the message should be routed, formatted, prioritized, and delivered.

A SmartMessage (a message constructed using the MessageML vocabulary) is an electronic message that provides a single, standard envelope to deliver its content to a host of communications devices and applications, referred to in this document as target

communications devices or endpoints. These endpoints include mobile phones with messaging capability, wireless PDAs, pagers, fax machines, PC email inboxes, instant messaging applications, and standard telephones. The MessageML architecture separates the content from the presentation, and allows the sender to define how its message content should be displayed to the user on these various endpoints. The Informant (a sender of a MessageML message) codes the message in XML only once, and that same message would be sent to a pager or an email account in HTML format, or to any other communications device, depending on the recipient's preferences, and displayed properly on each of these devices.

The present invention works in conjunction with communications platforms, and in particular, the communications platform known as a "MessageML Processing Platform", of which the "Centerpost Platform" is one example of this type of platform. When used in conjunction with a communications platform, the novel electronic messaging system enables individuals to receive, store, synthesize, format and intelligently process XML-based electronic messages and standard SMTP email from corporations or any other senders in a way that fully integrates individuals' existing email boxes, cell phones, pagers, fax machines, telephones, and other devices. This novel electronic messaging system used in conjunction with a communications platform reduces complexity, lowers costs, and increases capabilities for corporations that interact electronically with their customers.

The system along with the communications platform enables many valuable notification applications for corporations such as airlines, brokerages, financial institutions, and retailers, among others. For individuals, the present invention delivers valuable information while simplifying the management of all electronic communications. The communications platform, used in conjunction with the present invention, enables corporations and third parties to define and create applications using a common, centralized platform. The novel system can be used 1) by individuals via a web-mail like offering, 2) by telecommunications and internet service providers on a hosted or licensed basis, 3) by corporations on a hosted or licensed basis, or 4) by other service providers on a hosted or licensed basis.

Figure 3A shows a schematic view of a communications platform utilizing certa aspects of the present invention, to allow the transmission of electronic messages between corporations and individuals. This novel system along with the communicatior platform is also intended to function between corporations, and between individuals, anc is not merely limited to communications between corporations and individuals. Figure 3B shows some examples of the flow of communication using such a centralized, user-centric hub, along with the novel system described herein. Figure 3C lists some of the example interactions using the communications platform along with the electronic messaging system of the present invention.

Detailed Description of the Figures

Figure 1 is a block diagram detailing the need for a novel communications platform capable of utilizing intelligent messages and integrating endpoints;

Figure 2 is a schematic diagram displaying electronic messaging without the novel communications platform;

Figures 3A through 3C are block diagrams and charts displaying and indicating electronic messaging with the novel communications platform, along with examples of interactions available due to the platform;

Figure 4 is a block diagram illustrating the interrelationship between the elements of the SmartMessage of the present invention;

Figure 5 is a block diagram illustrating the interrelationship between the SmartMessage, the Stylesheets, and the activity and event payloads of the present invention;

Figure 6 is a block diagram illustrating the interrelationship between activity classes and event classes of the present invention;

Figure 7 is a block diagram illustrating the formatting and routing concept of the present invention;

Figure 8 is a block diagram illustrating a high-level perspective of the entities involved in the SmartMessage architecture of the present invention;

Figure 9 is a block diagram illustrating the interrelationship between the different entities of the present invention;

Figure 10 is a block diagram illustrating the interrelationship between the elements of the SmartMessage of the present invention, including the receipt function; and

Figure 11 is a diagram illustrating the flow of the SmartMessage of the present invention, including the receipt function.

Detailed Description of the Invention

The novel electronic messaging system or MessageML system is a new type of electronic messaging. It provides a configuration whereby message content contains information that drives how the message is processed. This means that even before a user reads a message, it has been prioritized, formatted, and routed to a pre-defined communications device or application. The sender of the message creates the actual content of the message and, using the MessageML vocabulary, tags the content with meaning to enable intelligent processing of the message based on the meaning of the content. MessageML tags can be used to define message attributes, associate multiple messages with each other, allow precise message formatting, enable the definition and publishing of message classes, define possible responses and how a reply should be delivered, enable and/or disable specific endpoint types, provide a security layer, pre-configure users and endpoints, extensibly tag raw content, define stylesheets, define messaging handling characteristics (e.g., message expiration), define message senders, define message receipt notifications, etc. The MessageML system is based upon the eXtensible Markup Language (XML).

The system is a precise XML vocabulary that can be utilized by a messaging platform to process messages based on the meaning of their content and interpretation of the accompanying messaging vocabulary. The system provides a configuration whereby the message content contains information to, among other things, determine how the message is processed. The sender of the message creates the actual content of the

message and uses the MessageML vocabulary to tag the content with meaning so that the message may be properly processed.

MessageML is a new electronic communications standard that applies XML, the widely accepted language of e-commerce, to messaging. MessageML is a precise XML vocabulary applied to messages to individuals. Unlike emails based on the current SMTP standard, which simply contain text content, MessageML messages contain a set of self-descriptive attributes that convey what the message's content means. Processing engines can use these attributes to intelligently manipulate and process the message, permitting advanced functionality such as routing of messages to any electronic device or messaging account; automatic formatting for various communications devices; automatic self-organization of messages within inboxes; and numerous other applications.

MessageML messages contain embedded XML tags that describe certain attributes of that message. A MessageML message processing platform matches the attributes of an incoming MessageML message with information about a recipient's communication devices and how they are used to intelligently route, format, and deliver the message. More specifically, messages composed using the MessageML vocabulary and architecture contain raw data and XML tags that describe that message's particular attributes, such as its reach, immediacy, sensitivity, content, expiration, and context. The message processor examines this meta-data in processing the message. The message processor also interprets meta-data about the sender and about the type of message, which are contained in stylesheets residing on the sender's server. The sender attributes comprise items such as the sender's name, website address, and industry category, while the message type attributes include the type of activity the message represents (e.g. Travel Itinerary) and how those messages should be displayed on various communications devices. By applying the sender's stylesheets to the smart message itself and factoring in the recipient's device preferences, the message processor determines how the message should be routed, formatted, prioritized, and delivered.

The XML standard is defined by the W3C (http://www.w3.org/XML/Schema), a consortium created in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability.

The standard provides a means for defining the structure, content and semantics of XML documents.

Figure 4 illustrates the relationship between the elements comprising the MessageML design. The SmartMessage 10 is central to the entire architecture.

A SmartMessage 10 is an electronic message that provides a single, standard envelope to deliver its content to a host of communications devices and applications, referred to in this document as target communication devices or endpoints 12. These endpoints 12 include mobile phones with messaging capability, wireless PDAs, pagers, fax machines, PC email inboxes, instant messaging applications, and standard telephones. Thus, in terms of its XML coding, a SmartMessage 10 would look the same regardless of whether it was sent to a pager or to an email account in HTML format. The MessageML architecture separates the content from the presentation, and allows the sender to define how its message content should be displayed to the user on these various endpoints 12.

A SmartMessage 10 also contains a robust set of self-descriptive attributes or meta-data about its content. By examining this meta-data, a processing platform or recipient can determine how the message should be handled, for example, how the message should be prioritized, handled, formatted, and delivered.

The MessageML system also incorporates message stylesheets that are applied to the message content. These stylesheets are used to create communication-device specific message formats. By separating the content from the presentation, the contents of a SmartMessage 10 can be transformed to any other presentation format. This allows the MessageML system to inter-operate and leverage other proprietary and XML-based communications formats. For example, a SmartMessage 10 could be sent to a WAP-enabled phone by transforming its contents to WML or a speech recognition dialog could be initiated over a telephone by transforming a SmartMessage 10 into a VoiceXML document.

Accordingly, it becomes easier for both the MessageML Service Provider and their recipients to manage, receive, route, and interact with electronic messages.

As evidenced from the above process flow, there are many different elements and entities involved in delivering a SmartMessage 10. As described above, the

SmartMessage document 10 is the main element of the MessageML system or architecture. In simple terms, a SmartMessage 10 is a well-formed XML document with a robust set of attributes that define and describe the embedded message content, which itself is a well-formed XML document. MessageML refers to its message content as its payload. Figure 5 illustrates the interrelationship between the SmartMessage 10, the Stylesheets, and the activity 20 and event 22 payloads.

The SmartMessage 10 itself has no meaningful interpretation without some type of definition. The definition is needed so that the recipient of the message can make decisions about how to handle the message before they read it. For example, the MessageML system provides the necessary constructs to provide certain information to the recipient even before the message is opened, i.e., whom the message is from, whether or not it is a flight cancellation, bank overdraft notification, etc., where to route the message, and what format to display itself.

MessageML definitions are created prior to sending a SmartMessage 10 and the definitions reside in the Informant Stylesheet 14 and SmartMessage Stylesheet 16. The SmartMessage 10 has a reference to the appropriate Informant Stylesheet 14 and SmartMessage Stylesheet 16. As described below, this association identifies the SmartMessage 10 with a specific Informant, activity class, event class, and validating XML schemas.

The first step in the SmartMessage process is for the sender to create an Informant Stylesheet 14. The Informant Stylesheet 14 defines meta-data about the Informant and the Informant's valid sources or locations from where its associated SmartMessages 10 can originate. This document is stored on the Informant's web server (not shown). Next, the Informant creates a SmartMessage Stylesheet 16 for each type of activity class it wishes to deliver to its recipients. For example, the Informant may create a SmartMessage Stylesheet 16 for travel, and could entitle this activity class "Travel Itinerary". Within each activity class is a collection of event classes (or message types) associated with that activity class, so "Flight Cancellation", "Itinerary Change", etc. could be events within the "Travel Itinerary" activity. The Informant defines what the message content or payload looks like for each event class by defining their XML

schemas, which are embedded within the SmartMessage Stylesheet 16. XSL documents are also created and embedded into the SmartMessage Stylesheet 16 to define how each event's payload should be displayed to the various endpoints 12. This document is also stored on the Informant's web server.

The MessageML system users sign up to receive SmartMessages 10 from an Informant with a MessageML Service Provider 18. Each SmartMessage user decides where the Informant's various SmartMessage event classes are to be delivered, specifically to which of the user's endpoints 12. The Informant then creates and sends a SmartMessage 10 to a set of SmartMessage user accounts hosted by this MessageML Service Provider 18. The MessageML Service Provider 18 receives, processes and delivers the SmartMessage 10 to the specified endpoint 12 of each addressed user.

The Informant Stylesheet 14 defines information about the Informant or sender of the information and its valid transport sources or locations from where it will send its messages. For each source the allowable Internet access protocol is also defined. Further, the SmartMessage Stylesheet 16 defines the message payload's meta-data and structure, as well as how it should be rendered using XSL for the various endpoints 12.

The main role of the Informant Stylesheet 14 is to provide information about and authenticate the sender of the message. The Informant Stylesheet 14 also contains meta-data about the Informant itself, such as its name, website address, industry category, etc. When a MessageML Service Provider 18 receives a SmartMessage 10 from a sender, the Service Provider can check the physical IP address, domain, or SMTP email account from which the SmartMessage 10 was received. If the source address matches one of the entries in the Informant Stylesheet 14, it is considered authentic, otherwise, the message is rejected.

The Informant Stylesheet 14 usually resides on the Informant's web server and is versioned by its filename. For example, the Informant Stylesheet http://smartmessage.messageml.org/stylesheets/informant/v1-0.xml describes version 1.0 of the document.

The SmartMessage Stylesheet 16 serves three main functions. First, it defines activity and event class meta-data and organization. Second, it defines the XML schemas

of the activity 24 and event class 26 payloads or content 20, 22. Third, it defines how the payload 20, 22 is rendered via XSL for specific endpoints12.

As discussed above, an activity class 24 is a grouping of event classes 26. For example, the activity class 24 called Travel Itinerary would have several event classes 26 associated with it. These may include flight cancellations, flight changes, rental car and hotel confirmations, and the like. An event class 26 can belong to one and only one activity class 24. Figure 6 illustrates the relationship between activity classes 24 and event classes 26.

Activity classes 24 and event classes 26 also have meta-data describing their duration and frequency. This information can be used to understand the timeliness of the message itself. Also, an event class 26 has additional meta-data about its purpose, sensitivity, reach, immediacy, and category. This information can be used to filter SmartMessages 10 and specify routing rules and instructions.

When an update is needed, all new SmartMessages 10 would change their reference to point at the new SmartMessage Stylesheet version, for example, from itinerary-v1-0.xml to itinerary-v1-1.xml. This design maintains backward compatibility.

The XML schemas define the structure of the SmartMessage payload. These XML schemas are embedded within the SmartMessage Stylesheet 16 in the <activity-payload-schema> and <event-payload-schema> elements. These schemas follow the XML Schema format as defined by the W3C, as described above. The activity payload 20 describes information common to all its event classes 26, and event payload 22 describes information specific to that event instance.

Examples of SmartMessage Schemas (SmartMessage 10, SmartMessage Stylesheet 16, and Informant Stylesheet14) are as follows:

SmartMessage:

```
<?xml version="1.0"?>
<!-- smartmessage schema -->
<Schema name="v1-1.xdr" xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes">
    <ElementType name="smXML" content="eltOnly" order="seq">
```

```
<AttributeType name="smartmessage-date" dt:type="dateTime" required="yes"/>
<AttributeType name="smartmessage-stylesheet-class" dt:type="string"
    required="yes"/>
<AttributeType name="informant-stylesheet-version" dt:type="string"
    required="yes"/>
<AttributeType name="smartmessage-id" dt:type="string" required="yes"/>
<AttributeType name="protocol-version" dt:type="string" required="yes"/>
<AttributeType name="smartmessage-stylesheet-version" dt:type="string"
    required="yes"/>
<AttributeType name="informant-stylesheet-class" dt:type="string"
    required="yes"/>
<attribute type="smartmessage-date"/>
<attribute type="smartmessage-stylesheet-class"/>
<attribute type="informant-stylesheet-version"/>
<attribute type="smartmessage-id"/>
<attribute type="protocol-version"/>
<attribute type="smartmessage-stylesheet-version"/>
<attribute type="informant-stylesheet-class"/>
<element type="route"/>
<element type="activity"/>
<element type="event"/>
</ElementType>
<ElementType name="route" content="eltOnly" order="seq">
<element type="from"/>
<element type="to" minOccurs="1" maxOccurs="*"/>
<element type="receipt-request" minOccurs="0" maxOccurs="*"/>
</ElementType>
<ElementType name="from" content="empty">
<AttributeType name="reply-address" dt:type="string"/>
<AttributeType name="reply-protocol" dt:type="enumeration"
    dt:values="http smtp" default="smtp"/>
<AttributeType name="from-address" dt:type="string" required="yes"/>
<attribute type="reply-address"/>
<attribute type="reply-protocol"/>
<attribute type="from-address"/>
</ElementType>
<ElementType name="to" content="empty">
<AttributeType name="to-type" dt:type="enumeration" dt:values="to cc bcc"
    default="to"/>
<AttributeType name="to-protocol" dt:type="enumeration" dt:values="http smtp"
    default="smtp"/>
<AttributeType name="to-address" dt:type="string" required="yes"/>
<attribute type="to-type"/>
<attribute type="to-protocol"/>
<attribute type="to-address"/>
```

```
</ElementType>
<ElementType name="receipt-request" content="empty">
    <AttributeType name="receipt-type" dt:type="enumeration"
        dt: values="ack nak retry" default="nak"/>
    <AttributeType name="receipt-protocol" dt:type="enumeration"
        dt:values="http smtp" default="smtp"/>
    <AttributeType name="receipt-event" dt:type="string"/>
    <AttributeType name="receipt-address" dt:type="string" required="yes"/>
    <attribute type="receipt-type"/>
    <attribute type="receipt-protocol"/>
    <attribute type="receipt-event"/>
    <attribute type="receipt-address"/>
</ElementType>
<ElementType name="activity" content="eltOnly" order="seq">
    <AttributeType name="activity-title" dt:type="string" required="yes"/>
    <AttributeType name="activity-id" dt:type="string" required="yes"/>
    <AttributeType name="closed-date" dt:type="dateTime" required="yes"/>
    <AttributeType name="activity-class" dt:type="string" required="yes"/>
    <AttributeType name="activity-status" dt:type="string" required="yes"/>
    <AttributeType name="activity-url" dt:type="string" required="yes"/>
    <attribute type="activity-title"/>
    <attribute type="activity-id"/>
    <attribute type="closed-date"/>
    <attribute type="activity-class"/>
    <attribute type="activity-status"/>
    <attribute type="activity-url"/>
    <element type="activity-payload"/>
</ElementType>
<ElementType name="activity-payload" content="eltOnly" order="seq"/>
<ElementType name="event" content="eltOnly" order="seq">
    <AttributeType name="event-description" dt:type="string" required="yes"/>
    <AttributeType name="event-class" dt:type="string" required="yes"/>
    <AttributeType name="event-id" dt:type="string" required="yes"/>
    <AttributeType name="event-url" dt:type="string" required="yes"/>
    <attribute type="event-description"/>
    <attribute type="event-class"/>
    <attribute type="event-id"/>
    <attribute type="event-url"/>
    <element type="event-payload"/>
</ElementType>
<ElementType name="event-payload" content="eltOnly" order="seq"/>
</Schema>
```

SmartMessage Stylesheet:

```
<?xml version="1.0"?>
<!-- smartmessage stylesheet -->
<Schema name="v1-1.xdr" xmlns="urn:schemas-microsoft-com:xml-data"
        xmlns:dt="urn:schemas-microsoft-com:datatypes">
    <ElementType name="smSmartMessageStylesheet" content="eltOnly" order="seq">
        <AttributeType name="smartmessage-stylesheet-class" dt:type="string"
                required="yes"/>
        <AttributeType name="smartmessage-stylesheet-version" dt:type="string"
                required="yes"/>
        <attribute type="smartmessage-stylesheet-class"/>
        <attribute type="smartmessage-stylesheet-version"/>
        <element type="activity-class" minOccurs="1" maxOccurs="*"/>
    </ElementType>
    <ElementType name="activity-class" content="eltOnly" order="seq">
        <AttributeType name="activity-name" dt:type="string" required="yes"/>
        <AttributeType name="activity-duration" dt:type="enumeration"
                dt:values="hours days weeks months year ongoing" required="yes"/>
        <AttributeType name="event-frequency" dt:type="enumeration"
                dt:values="hours daily weekly monthly once" required="yes"/>
        <attribute type="activity-name"/>
        <attribute type="activity-duration"/>
        <attribute type="event-frequency"/>
        <element type="activity-payload-schema" minOccurs="0" maxOccurs="1"/>
        <element type="activity-xsl-default"/>
        <element type="activity-xsl-endpoint" minOccurs="0" maxOccurs="*"/>
        <element type="event-class" minOccurs="1" maxOccurs="*"/>
    </ElementType>
    <ElementType name="activity-payload-schema" content="eltOnly" order="seq"/>
    <ElementType name="activity-xsl-default" content="eltOnly" order="seq"/>
    <ElementType name="activity-xsl-endpoint" content="eltOnly" order="seq">
        <AttributeType name="endpoint-type" dt:type="enumeration"
                dt:values="browser html-email text-email tiny-email fax voice-phone
                instant-message" required="yes"/>
        <attribute type="endpoint-type"/>
    </ElementType>
    <ElementType name="event-class" content="eltOnly" order="seq">
        <AttributeType name="frequency" dt:type="enumeration"
                dt:values="hourly daily weekly monthly once" default="once"/>
        <AttributeType name="sensitivity" dt:type="enumeration"
                dt:values="private normal public" default="normal"/>
        <AttributeType name="reach" dt:type="enumeration"
                dt:values="broadcast group individual" default="individual"/>
        <AttributeType name="purpose" dt:type="enumeration"
                dt:values="question information instruction" default="information"/>
        <AttributeType name="immediacy" dt:type="enumeration"
```

```
            dt:values="minutes hours days weeks months" default="days"/>
        <AttributeType name="category" dt:type="enumeration"
            dt:values="auctions banking career community credit-card education
            entertainment general health information investing miscellaneous personal
            shopping system telemetry travel" default="miscellaneous"/>
        <AttributeType name="event-name" dt:type="string" required="yes"/>
        <attribute type="frequency"/>
        <attribute type="sensitivity"/>
        <attribute type="reach"/>
        <attribute type="purpose"/>
        <attribute type="immediacy"/>
        <attribute type="category"/>
        <attribute type="event-name"/>
        <element type="event-payload-schema" minOccurs="0" maxOccurs="1"/>
        <element type="event-xsl-default"/>
        <element type="event-xsl-endpoint" minOccurs="0" maxOccurs="*"/>
    </ElementType>
    <ElementType name="event-payload-schema" content="eltOnly" order="seq"/>
    <ElementType name="event-xsl-default" content="eltOnly" order="seq"/>
    <ElementType name="event-xsl-endpoint" content="eltOnly" order="seq">
        <AttributeType name="endpoint-type" dt:type="enumeration"
            dt:values="browser html-email text-email tiny-email fax voice-phone
            instant-message" required="yes"/>
        <attribute type="endpoint-type"/>
    </ElementType>
</Schema>
```

Informant Stylesheet:

```
<?xml version="1.0"?>
<!-- informant stylesheet schema -->
<Schema name="v1-1.xdr" xmlns="urn:schemas-microsoft-com:xml-data"
    xmlns:dt="urn:schemas-microsoft-com:datatypes">
    <ElementType name="smInformantStylesheet" content="eltOnly" order="seq">
        <AttributeType name="logo-url" dt:type="string"/>
        <AttributeType name="signup-url" dt:type="string"/>
        <AttributeType name="informant-stylesheet-version" dt:type="string"
            required="yes"/>
        <AttributeType name="home-url" dt:type="string"/>
        <AttributeType name="informant-category" dt:type="enumeration"
            dt:values="auctions banking career community credit-card education
            entertainment general health information investing miscellaneous personal
            shopping system telemetry travel" required="yes"/>
        <AttributeType name="informant-description" dt:type="string" required="yes"/>
        <AttributeType name="informant-stylesheet-class" dt:type="string"
```

```
            required="yes"/>
        <AttributeType name="informant-name" dt:type="string" required="yes"/>
        <attribute type="logo-url"/>
        <attribute type="signup-url"/>
        <attribute type="informant-stylesheet-version"/>
        <attribute type="home-url"/>
        <attribute type="informant-category"/>
        <attribute type="informant-description"/>
        <attribute type="informant-stylesheet-class"/>
        <attribute type="informant-name"/>
        <element type="valid-transport-source" minOccurs="1" maxOccurs="*"/>
    </ElementType>
    <ElementType name="valid-transport-source" content="textOnly">
        <AttributeType name="transport-protocol" dt:type="enumeration"
            dt:values="smtp http" default="smtp"/>
        <AttributeType name="transport-source" dt:type="string" required="yes"/>
        <attribute type="transport-protocol"/>
        <attribute type="transport-source"/>
    </ElementType>
</Schema>
```


As described herein, a SmartMessage 10 is an XML document sent by an Informant to a SmartMessaging account hosted by a MessageML Service Provider 18. A SmartMessage 10 carries Event content or a payload 22, such as a flight cancellation or shipment confirmation, and is associated with and updates an Activity 24, such as a Travel Itinerary or DVD order.

Meta-data, such as formatting information, is found by referencing the associated Informant Stylesheet 14 (which describes the Informant) and the associated SmartMessage Stylesheet 16 (which describes the activity 24 and event 26). The structure, elements and attributes of a SmartMessage document 10 include smXML 28, route, from, to, receipt-request, activity, activity-payload, event, and event-payload. The structure, elements and attributes are summarized in the following table.

Element and Level	Opt	Multi	Attributes	Sample Value (underline=default)
smXML				

			protocol-version	1.1 (current version)
			SmartMessage-id	123ABC456DEFfutureair.com (any globally unique ID)
			SmartMessage-date	2000-03-17T15:10:33-6:00
			informant-stylesheet-class	http://smartmessage.futureair.com/stylesheets/informant/
			informant-stylesheet-version	Informant-v1-2.xml
			smartmessage-stylesheet-class	http://smartmessage.futureair.com/, http://smartmessage.messageml.org/
			smartmessage-stylesheet-version	smartmessage-v2-3.xml
route				
from				
			from-address	informant@futureair.com
	X		reply-protocol	http, smtp
	X		reply-address	sm@futureair.com
To		X		
			to-address	customer@hotmail.com
	X		to-type	to, cc, bcc
	X		to-protocol	http, smtp
receipt-request	X	X		
	X		receipt-type	ack, nak, retry
	X		receipt-event	Received, processed, delivery-status
	X		receipt-protocol	http, smtp
			receipt-address	receipts@futureair.com
activity				
			activity-class	Traveliteinerary, Order, BillingCycle,
			activity-id	(an ID to uniquely identify the activity instance)
			activity-url	http://www.futureair.com/flightitnerary.cgi?id=6576782345
			activity-title	March 29 Travel Itinerary
			activity-status	Flight reservations confirmed
			closed-date	(date activity was closed, blank if open, can be postdated)
Activity-payload	X			(a well formed XML document w/a single root)
Event				
			event-class	FlightCancellation, Shipment, PaymentReceived
			event-id	(uniqueID to define event in corresponding activity)

	event-url	http://www.futureair.com/flightstatus.cgi?flight=FA234
	event-description	Flight FA234 has been cancelled
Event-payload		(a well formed XML document w/a single root)

The "smXML" element 28 is the root element of the SmartMessage document 10, and includes the following attributes.

Attributes	Purpose
protocol-version	The SmartMessage protocol version. The SmartMessage XML parser uses this information to maintain version specific functionality.
SmartMessage-id	A globally unique ID used to identify the SmartMessage.
SmartMessage-date	The date and time of SmartMessage submission in ISO 8601 format. The format is YYYY-MM-DDThh:mm:ss-hh:mm (for example, 1997-07-16T19:20:30+01:00).
informant-stylesheet-class	The Internet location of the associated Informant Stylesheet. This is the complete URL of the website where the Informant stylesheet is stored. For example, "http://smartmessage.messageml.org/stylesheets/informant/". Note, the ending "/" must be present for the URL to be considered complete.
informant-stylesheet-version	The version of the Informant Stylesheet XML document, which contains the metadata about an Informant. This is a file located at the *informant-stylesheet-class*. For example, if the *informant-stylesheet-class* is "http://smartmessage.messageml.org/stylesheets/informant/" and the *informant-stylesheet-version* is "v1-0.xml", then the SmartMessage processor will resolved the complete location of the Informant stylesheet to "http://smartmessage.messageml.org/stylesheets/informant/v1-0.xml".
SmartMessage-stylesheet-class	The Internet location of the associated SmartMessage Stylesheet. This is the complete URL of the website where the SmartMessage

	stylesheet is stored. For example, "http://smartmessage.messageml.org/stylesheets/receipts/". Note, the ending "/" must be present for the URL to be considered complete.
SmartMessage-stylesheet-version	The version of the SmartMessage Stylesheet XML document, which contains the metadata about a SmartMessage. This is a file located at the *smartmessage-stylesheet-class*. For example, if the *smartmessage-stylesheet-class* is "http://smartmessage.messageml.org/stylesheets/receipts/" and the *smartmessage-stylesheet-version* is "v1-0.xml", then the SmartMessage processor will resolved the complete location of the Informant stylesheet to "http://smartmessage.messageml.org/stylesheets/receipts/v1-0.xml".

The "route" element describes information about the SmartMessage recipients, the Informant and receipt requests. This is a container element.

The "from" element defines the sender of the SmartMessage 10 and message reply options, and includes the following attributes.

5

attributes	Purpose
from-address	The Informant's address in *'name@domain'* format. This address is populated by the Informant to describe.
reply-address (optional)	The Informant's reply address. This will be different based on the value of the reply-protocol attribute. When *smtp* is specified for the reply-protocol, the address is the Informant's valid SMTP e-mail account in *'name@domain'* format. When *http* is specified for the reply-protocol, the address is a valid URL, which will receive the reply via the HTTP POST method.
reply-protocol (optional)	The protocol in which the reply will be sent. Valid values are *http* and *smtp*. Default value is *smtp*. If *smtp* is specified an SMTP e-mail will be sent. If *http* is specified the reply will be posted via the HTTP POST

	method to the reply-address, which must be a valid URL.

The "to" element defines the recipients of the SmartMessage. This element occurs multiple times, once for each recipient of the SmartMessage 10, and includes the following attributes.

attributes	Purpose
to-type (optional)	The SMTP delivery method of the message. This specifies how the message should be addressed to the Recipient when *smtp* is specified for the to-protocol attribute of this element. Valid values are *to*, *cc*, *bcc*. Default value is *to*.
to-protocol (optional)	The protocol in which the SmartMessage will be sent. Valid values are *http* and *smtp*. Default value is *smtp*. If *smtp* is specified an SMTP e-mail will be sent. If *http* is specified the message will be posted via the HTTP POST method to the to-address, which must be a valid URL.
to-address	When *smtp* is specified for the to-protocol, the address is the recipient's valid SmartMessage account in '*name@domain*' format. This address is returned to a recipient from the MessageML Service Provider when he completes the registration process. When *http* is specified for the to-protocol, the address is a valid URL, which will receive the SmartMessage via the HTTP POST method.

The "receipt-request element is optional and specifies that a receipt notification of the SmartMessage 10 be sent back to the sender or Informant. The element may be defined multiple times, once for each type of receipt request, and includes the following attributes.

Attributes	Purpose
receipt-type (optional)	Specifies the acknowledgement type of receipt to be sent to the receipt address. The SmartMessage can send a success (ack), failure (nak), or retry attempt (retry) receipt notifications.

	Valid values are *ack*, *nak*, and *retry*. Default value is *nak*.
receipt-event (optional)	Specifies the type of receipt to be sent to the receipt-address. The receipt-address can be notified when the SmartMessage is either received by the SmartMessage Processor, has been processed by the SmartMessage Processor, or has been delivered to the Recipient. Valid values are *received*, *processed*, *delivery-status*. Default value is *processed*. When *delivery-status* is specified the receipt-address will receive a receipt for every retry attempt (retry), as well as the final delivery outcome: ack or nak.
receipt-protocol (optional)	The protocol in which the receipt will be sent. Valid values are *http* and *smtp*. Default value is *smtp*. If *smtp* is specified an SMTP e-mail will be sent with the receipt SmartMessage attached. If *http* is specified the receipt SmartMessage will be posted via the HTTP POST method to the receipt-address, which must be a valid URL including web page.
receipt-address (optional)	When *smtp* is specified for the receipt-protocol, the address is a valid SMTP e-mail account in '*name@domain*' format. When *http* is specified for the receipt-protocol, the address is a valid URL, which will receive the receipt via the HTTP POST method.

The "activity" element provides information about the activity associated with the SmartMessage 10. An activity is a categorization of events. For example, an activity may be "Travel Itinerary" with such related events as "Flight Cancellation", "Flight Arrival Time", etc., and includes the following attributes.

Attributes	purpose
activity-class	The class name of the activity. This activity class name must be defined by the Informant and must match a valid activity class (activity-class-name attribute) in the referenced SmartMessage Stylesheet (<smXML>\smartmessage-stylesheet-version). The matching activity-class-name in the SmartMessage

	Stylesheet contains metadata, the XML schema and endpoint specific XSL documents about the activity and more importantly defines the valid event classes for the activity class itself.
activity-id	A unique ID to identify the activity instance. Successive SmartMessages received with the same activity ID will be grouped by this ID. For example, multiple event classes like "Flight Arrival Time", "Flight Cancellation", etc. can be grouped under a single activity class called "01/01/200 Travel Itinerary" by associating the event classes with that ID.
activity-url	An associated URL to the activity. This may simply link back to the Informants web site or may contain query strings in the URL to link back to the recipient's account page.
activity-title	A descriptive title of the activity.
activity-status	A short description of the activity status.
closed-date	Date the activity was closed in ISO 8601 format. This date can be post-dated. If this attribute is empty the activity is assumed to be open.

The "activity-payload" element 20 is a well-formed XML document with a single root, embedded in the SmartMessage 10. This element contains information related to the activity. This payload's XML Schema is defined by the referenced SmartMessage Stylesheet 16, as the following examples shows.

```
<travel-itinerary xmlns="travel">
    <name>FutureAirlines Travel Itinerary<name>
    <description>Boston to Chicago with Hotel, Rental Car<description>
</travel-itinerary>
```

The "event" element provides information about the event associated with the activity specified in the activity element. An event is a specific instance of some type of SmartMessage 10 occurrence, for example a specific flight delay announcement, etc., and includes the following attributes.

Attributes	Purpose
event-class	The class name of the event. This event name must be

	defined by the Informant and match a defined event class (event-class-name attribute) in the referenced SmartMessage Stylesheet (<smXML>\smartmessage-stylesheet-version). This event-class must be also match a defined activity class in the SmartMessage Stylesheet. For example an airline type Informant may define "Flight Arrival Time", "Flight Cancellation", etc. event classes under a "Travel Itinerary" activity-class. The matching event-class-name in the SmartMessage Stylesheet contains metadata, the XML schema and endpoint specific XSL documents about the event. The event-payload will contain the information specific to the SmartMessage's instance of the event.
event-id	A unique ID to define the event in the corresponding activity. If another SmartMessage is sent with the same event ID then that SmartMessage will overwrite the existing instance.
event-url	An associated URL to the event. This may simply link back to the Informants web site or may contain query strings in the URL to provide more functionality.
event-description	A textual description of the event.

The "event-payload" 22 is a well-formed XML document with a single root containing information related to the event. This payload element's XML Schema is defined by the referenced SmartMessage Stylesheet 16, as the following example shows.

```
<flightcancel xmlns="flightcancel">
    <name>John Smith</name>
    <airline>FutureAirlines</airline>
    <destname>Atlanta, GA</destname>
    <destcode>ATL</destcode>
    <departname>Chicago, IL</departname>
    <departcode>ORD</departcode>
    <departtime>7:30pm</departtime>
    <departdate>6/29/2000</departdate>
```

```
<flightnum>219</flightnum>
<newflight>999</newflight>
<newtime>11:50pm</newtime>
<newdate>6/29/2000</newdate>
<reason>Cancellation due to bad weather.</reason>
<customerservice>800-555-5555</customerservice>
</flightcancel>
```

As described herein, the Informant stylesheet 14 describes the meta-data about the Informant, such as logo and description, and valid transport sources. The Informant stylesheet 14 resides on one of the Informant's web servers. A MessageML Processing Platform hosted by a MessageML Service Provider 18 caches these style sheets in its own database so it can refer to them quickly without incurring another roundtrip over the Internet.

The structure, elements, and attributes of an Informant Stylesheet 14 document are summarized in the following table.

Element	Opt	Multi	Attributes	Sample Value (underline=default)
SmInformantStylesheet				
			informant-name	FutureAir
			informant-stylesheet-class	http://sm.futureair.com
			informant-stylesheet-version	Informant-v1-2.xml
			informant-description	Amazon.com, earth's largest store, featuring over ...
			logo-url	http://sm.amazon.com/Logo-v2-3.jpg (should be 50x30 pixels)
			home-url	http://www.amazon.com/
			signup-url	http://www.amazon.com/sm-signup.asp
			informant-category	auctions, banking, career, community, credit-card, education, entertainment, general, health, information, investing, miscellaneous, personal, shopping,

			system, telemetry, travel
valid-transport-source	X		
		transport-protocol	smtp, http
		transport-source	*@amazon.com, 38.240.27.*, (wildcard * allowed)

The "smInformantStylesheet" element is the root element of this document, and includes the following attributes.

Attributes	Purpose
informant-name	The name of the Informant.
informant-stylesheet-class	The Internet location of the Informant Stylesheet. This is the complete URL of the website where the Informant stylesheet is stored. For example, "http://smartmessage.messageml.org/stylesheets/informant/". Note, the ending "/" must be present for the URL to be considered complete. This attribute describes where the document is located on the Internet itself.
informant-stylesheet-version	The version of the Informant Stylesheet XML document. This is an XML file located at the *informant-stylesheet-class*. For example, if the *informant-stylesheet-class* is "http://smartmessage.messageml.org/stylesheets/informant/" and the *informant-stylesheet-version* is "v1-0.xml", then the SmartMessage processor will resolved the complete location of the Informant stylesheet to "http://smartmessage.messageml.org/stylesheets/informant/v1-0.xml"
informant-description	Description of the Informant. Possibly the informant's branded slogan, etc.
Logo-url	Informant's logo in JPG formant – should be 50x30 pixels in dimension – specified in a URL format.
Home-url	The URL of the home page of the Informant.
signup-url	A URL which provides a signup or user registration
informant-category	The category in which the informant belongs.

	Valid values are : *auctions, banking, career, community, credit-card, education, entertainment, general, health, information, investing, miscellaneous, personal, shopping,* system, *telemetry, travel*

The "valid-transport-source" element describes the valid sources from where SmartMessages 10 can originate for the specified Informant. An Informant can create different version of its Informant Stylesheet 14 to define different sets of valid transport sources. The element can occur multiple times, once for each valid transport, and includes the following attributes.

attributes	Purpose
transport-protocol	The protocol the Informant can send the SmartMessage from the source defined in the transport-source attribute. Valid values are *http* and *smtp*. Default value is *smtp*. If *smtp* is specified a SmartMessage can only be received by the specified transport-source in the form of *name@domain*. If *http* is specified a SmartMessage can only be received by the specified transport-source in the form of a valid IP address.
transport-source	Describes the valid sources for the associated transport-protocol. By associating itself with an Informant Stylesheet (<smXML>/informant-stylesheet-version) a SmartMessage can restrict its transport sources to a small, defined set of locations to prevent unauthorized submissions. For a transport-protocol of *smtp*, the format *name@domain* is used. The transport-source can be entered with a wildcard mask using an asterisk. For example, by specifying the source **@messageml.org* and protocol of *smtp* all user's with a domain name of messageml.org in their SMTP e-mail account will be considered valid transport sources. For a transport-protocol of *http*, the format is a valid IP address. An * can be used as a wildcard for any octet

	within the IP address. For example, by specifying the source *123.456.789.** and protocol of *http* all user's whose first three octets match 123.456.789 will be considered valid transport sources.

An example of a Informant Stylesheet 14, which describes an Informant with multiple valid transport sources is as follows.

```
<smInformantStylesheet
        xmlns="x-
    schema:http://sm.futureairlines.com/schemas/def/smInformantStylesheetSchema1
    .1.xdr"
        informant-name="FutureAirlines"
        informant-stylesheet-class=http://www.futureairlines.com/definitions/
        informant-stylesheet-version="faInformantStylesheet.xml"
        informant-description="FutureAirlines - The way you should fly"
        logo-url="http://www.futureairlines.com/futureairlines.jpg"
        home-url="http://www.futureairlines.com"
        informant-category="travel">
        <valid-transport-source transport-source="*@futureairlines.com"/>
        <valid-transport-source transport-source="*@travelocity.com"/>
        <valid-transport-source transport-protocol="http" transport-
    source="123.456.789.*"/>
        <valid-transport-source transport-protocol="http" transport-
    source="321.654.*.*"/>
    </smInformantStylesheet>
```

As described herein, a SmartMessage stylesheet 16 describes the meta-data about a SmartMessage 10, such as how to format it on different types of endpoints 12. The SmartMessage stylesheet 16 generally resides on one of the Informant's web servers, but Informants can also share a common stylesheet on a shared server like smartmessage.messageml.org.

A MessageML Processing Platform should cache these stylesheets in its own database so it can refer to them quickly without hitting the Internet. A single SmartMessage Stylesheet XML document 16 can contain meta-data about several SmartMessage activities 24 and events 26.

The structure, elements, and attributes of a SmartMessage Stylesheet document 16 are summarized in the following table.

Element	Opt	Multi	Attributes	Sample Value (underline=default)
smSmartMessageStylesheet				
			smartmessage-stylesheet-class	http://sm.amazon.com/, http://smartmessage.messageml.org/
			smartmessage-stylesheet-version	SmartMessage-v2-3.xml
Activity-class		X		
			activity-name	Order, BillingCycle, TravelItinerary
			activity-duration	hours, days, weeks, months, years, ongoing
			activity-frequency	hourly, daily, weekly, monthly, once
activity-payload-schema	X			(a well formed XML Schema document w/a single root - embedded)
activity-xsl-default				(a well formed XSL document w/a single root - embedded)
activity-xsl-endpoint	X	X		(a well formed XSL document w/a single root - embedded)
			endpoint-type	browser, html-email, text-email, tiny-email, fax, voice-phone, instant-message
event-class		X		
			event-name	Shipment, PaymentReceived, FlightCancellation
			Purpose	question, information, instruction
			Sensitivity	private, normal, public
			Reach	individual, group, broadcast
			Immediacy	minutes, hours, days, weeks, months
			Frequency	hourly, daily, weekly, monthly, once
			category	auctions, banking, career, community, credit-card, education, entertainment, general, health, information, investing, miscellaneous, personal, shopping,

				systems, telemetry, travel
event-payload-schema	X			(a well formed XML Schema document w/a single root - embedded)
event-xsl-default				(a well formed XSL document w/a single root - embedded)
event-xsl-endpoint	X	X		(a well formed XSL document w/a single root - embedded)
			endpoint-type	browser, html-email, text-email, tiny-email, fax, voice-phone, instant-message

The "smSmartMessageStylesheet" is the root element of this document, and includes the following elements.

Attributes	Purpose
smartmessage-stylesheet-class	The Internet location of the SmartMessage Stylesheet. This is the complete URL of the website where the SmartMessage Stylesheet is stored. For example, "http://smartmessage.messageml.org/stylesheets/receipts/". Note, the ending "/" must be present for the URL to be considered complete. This attribute describes where the document is located on the Internet.
smartmessage-stylesheet-version	The version of the SmartMessage Stylesheet XML document. This is an XML file located at the *smartmessage-stylesheet-class*. For example, if the *smartmessage-stylesheet-class* is "http://smartmessage.messageml.org/stylesheets/receipts/" and the *smartmessage-stylesheet-version* is "v1-0.xml", then the SmartMessage processor will resolve the complete location of the Informant stylesheet to "http://smartmessage.messageml.org/stylesheets/receipts/v1-0.xml" This attribute describes where the document is located on the Internet.

The "activity-class" element 24 is a predefined category used to group event classes 26. The activity class 24 is defined by the Informant for use in its own SmartMessages 10, and includes the following attributes.

Attributes	Purpose
activity-class-name	The name of the activity class. The activity-class attribute of the <activity> element in the SmartMessage must refers to a defined activity class in this document (see SmartMessage Definition section).
activity-duration	Describes the activities duration or life. Valid values are *hours, days, weeks, months, years, ongoing*
event-frequency	Describes how frequently events may occur within this activity. Valid values are *hourly, daily, weekly, monthly, once*

The "activity-payload-schema" is a well-formed XML schema document with a single root embedded within the document. This embedded XML Schema defines the structure for activity payload 20.

The "activity-xsl-default" is a well-formed XSL document with a single root embedded within the document. This embedded XSL document transforms the activity payload 20 (XML data) into an endpoint-independent presentation format. This definition is meant to be a "catch all" for rendering activity payload to an endpoint 12, if an endpoint specific XSL document 30 is not defined. Thus, the XSL transformation must be generic enough to be viewed on any endpoint.

<activity-xsl-endpoint>

The "activity-xsl-endpoint" is a well-formed XSL document with a single root embedded within the document, and contains the following attribute. This embedded XSL transforms the activity payload 20 (XML data) into an endpoint-dependent presentation format. The Informant can define this element for each supported endpoint device type 12. If an XSL is not defined for a valid endpoint device, then the activity-xsl-default will be used.

attributes	Purpose
endpoint-type	Describes the endpoint type that the embedded XSL document transforms to. An Informant can create a separate XSL for a specific endpoint device to generate the appropriate endpoint user interface. Valid values are browser, html-email, text-email, tiny-email, fax, voice-phone, instant-message

The "event-class" element 26 defines a classification of specific events related to an activity. For example, "Flight Cancellation" may be an event class 26 defined under the activity class 24 "Travel Itinerary," and includes the following attributes.

Attributes	Purpose
event-class-name	The name of the event class. The event-class attribute of the <event> element in the SmartMessage would refer to a defined event class in this document (see SmartMessage Definition section).
Purpose	Describes the purpose of the content with which the event is associated. Valid values are *question, information, instruction.* Default value is *information..*
sensitivity	Describes the sensitivity with which the event is associated. Valid values are *private, normal, public.* Default value is *normal.*
reach	Indicates whether the document was sent to a large public broadcast, to a specific group of individuals or to one individual. Valid values are *individual, group, broadcast.* Default value is *individual.*
immediacy	Sets the immediacy of the communication of this document payload. Dictates a measurement of how time sensitive is the document information. Information that is relatively 'old' after a few minutes should be set as 'minutes' immediacy (such as a traffic report) while information that is relative for a longer period of time is set to a higher value (such as a bill statement would be days or weeks).

	Valid values are *minutes, hours, days, weeks, months.* Default value is *days.*
frequency	Valid values are *hourly, daily, weekly, monthly, once.* Default value is *once.*

The "event-payload-schema" is a well-formed XML schema document with a single root embedded within the document. This embedded XML schema defines the structure for event content.

The "event-xsl-default" is a well-formed XSL document with a single root embedded within the document. This embedded XSL document transforms the event payload (XML data) into an endpoint-independent presentation format. This definition is meant to be a "catch all" for rendering the event payload to an endpoint, if an endpoint specific XSL document is not defined. Thus, the XSL transformation must be generic enough to be viewed on any endpoint.

The "event-xsl-endpoint" is a well-formed XSL document with a single root embedded within the document, and includes the following attributes. This embedded XSL document transforms the event payload 22 (XML data) into an endpoint-dependent presentation format. The Informant can define this element for each supported endpoint device type 12. If an XSL is not defined for a valid endpoint device 12, then the event-xsl-default will be used.

attributes	Purpose
endpoint-type	Describes the endpoint type that the embedded XSL document transforms to. An Informant can create a separate XSL for each endpoint device to generate the appropriate endpoint user interface. Valid values are *browser, html-email, text-email, tiny-email, fax, voice-phone, instant-message*

As described herein, SmartMessages 10 are transmitted to particular endpoints 12. XSL templates define how a SmartMessage 10 is rendered to a specific endpoint 12. The <activity-xsl-endpoint> and <event-xsl-endpoint> elements contain these XSL templates, which are embedded within the SmartMessage Stylesheet 16. A XSL template should be

created for each endpoint device 12. The <activity-xsl-default> and <event-xsl-default> XSL templates are required and are used to render the SmartMessage content when an endpoint specific XSL template is not defined. Figure 7 illustrates how XSL templates are embedded in the SmartMessage Stylesheet.

Since MessageML Stylesheets reside on the Informant's web server they will be retrieved at runtime for message processing. The MessageML Processor can optimize this operation by caching these Stylesheets once the first instance of a SmartMessage referring to the Stylesheets is sent. The Service Provider 18 can then present its users with this catalog of activities and events from which they can select prioritization and routing preferences.

SmartMessage documents 10 are validated against XML Schemas as defined above. Specifically, SmartMessage documents 10 make use of Microsoft's implementation of XML Schemas (http://msdn.microsoft.com/xml/reference/schema/start.asp).

MessageML's implemented XML schemas are stored on the MessageML Forum's website and define a versioned standard of the SmartMessage specification. For example, the XML schema defined for a SmartMessage Document is located at http://smartmessage.messageml.org/schemas/smartmessage/v1-1.xdr; the XML schema defined for a SmartMessage Stylesheet is located at http://smartmessage.messageml.org/schemas/stylesheets/smartmessage/v1-1.xdr; and the XML schema defined for a Informant Stylesheet is located at http://smartmessage.messageml.org/schemas/stylesheets/informant/v1-1.xdr

These XML schemas defined for MessageML should be referenced accordingly in the related SmartMessage documents in the XML namespace attribute. For example,

```
<smSmartMessageStylesheet
    xmlns="x-
    schema:http://smartmessage.messageml.org/schemas/stylesheets/smartmessage/v1
    -1.xdr"
    ...
```

<smInformantStylesheet
xmlns="x-schema:
http://smartmessage.messageml.org/schemas/stylesheets/informant/v1-1.xdr"
...

<smXML
xmlns="x-schema: http://smartmessage.messageml.org/schemas/smartmessage/v1-1.xdr"
...

Versioning is maintained at all levels of the SmartMessage architecture, including the protocol, XML schemas and SmartMessage documents 10. Documents are versioned through their file naming convention and their location or path, which describes its purpose. The term "Class" can be thought of as the path and the term "Version" can be thought of as the filename.

Examples of MessageML defined XML schemas include the SmartMessage Stylesheet XML schema, version v1-1.xdr, class http://smartmessage.messageml.org/schemas/stylesheets/smartmessage/; the Informant Stylesheet XML schema, version v1-1.xdr, class http://smartmessage.messageml.org/schemas/stylesheets/informant; the SmartMessage XML schema, version v1-1.xdr, class http://smartmessage.messageml.org/schemas/smartmessage/

Examples of MessageML XML documents include the Informant Stylesheet version v1-0.xml, class http://smartmessage.messageml.org/stylesheets/informant/, and the Receipts SmartMessage Stylesheet, version v1-0.xml, class http://smartmessage.messageml.org/stylesheets/receipts/.

The SmartMessage protocol version is also specified on the SmartMessage document 10 in the protocol-version attribute of the MessageML element. The current version is 1.1.

It is important to note that the SmartMessage Stylesheet 16 embeds several documents into a single document. This is done so that a set of documents can be easily versioned in their entirety. If, for example, these embedded documents were allowed to be referenced externally, then a change to one of the external documents would corrupt the versioning across the entire set. Thus, by consolidating the set of related document for a SmartMessage Stylesheet 16 into one, better version control is attained.

The users of a Service Provider implementing services based on MessageML sign up to receive SmartMessages from the Informant. Each user decides where the Informant's various SmartMessage event classes are to be delivered, specifically to which of the user's endpoints. The Informant then creates and sends a SmartMessage to a set of user accounts hosted by this MessageML Service Provider. The MessageML Service Provider receives, processes and delivers the SmartMessage to the specified endpoint of each addressed user.

Figure 8 illustrates a high-level perspective of the entities involved in the SmartMessage architecture 32. The Informant 34 creates the SmartMessage 10 from data on an information system. Informants 34 also create and host a set of SmartMessage Stylesheets 16 and Informant Stylesheets 14. The User 36 receives SmartMessages 10 through their SmartMessage account with a MessageML Service Provider 18. The SmartMessage Application Interface 38 provides an interface, either graphical or programmatic, to the SmartMessage Transfer Agent (SMTA) 40. This SmartMessage Application Interface can be thought of as the user application and the SMTA as the reusable code object.

The SmartMessage Transfer Agent (SMTA) 40 sends and receives SmartMessages 10 passing the message data to the SmartMessage Application. The SMTA 40 sends SmartMessages 10 through either the HTTP or SMTP Internet access protocols 42. The MessageML Service Provider (MMLSP) 18 provides the technology infrastructure and platform required to process and route SmartMessages 10 to end users or message recipients 36. This includes a web site to host the URL for HTTP access, mail servers for SMTP access, and a web application for Recipients to manage and configure their SmartMessage account.

Through a web application users register all their messaging or endpoint devices 12 on which they receive messages, like mobile phones, PDAs, pagers, fax machines, email inboxes, instant messaging, etc. The web application also serves as a message management tool providing a folio, or single repository, for a Recipient's SmartMessages 10.

The MMLSP 18 also maintains an endpoint gateway infrastructure to route SmartMessages 10 accordingly. By implementing a rules based processing module 44, SmartMessages 10 can be routed based on user preferences.

MessageML documents are transmitted to a MessageML Service Provider 18 over the Internet using the HTTP and SMTP protocols 42. By using these protocols 42, the MessageML system is able to utilize a wide base of already existing infrastructure for message communications.

Once processed, the MessageML Service Provider 18 delivers the message to the recipient via its endpoint gateway services 46, which may use voice, fax, paging, SMTP, HTTP, or other protocols. Figure 9 illustrates the interactions between the different MessageML 32 entities.

Using the HTTP POST method, an Informant can post a MessageML document 10 to a web server (page) hosted by a MessageML Service Provider 18 who in turn will process the message and route it to the specified recipients 36. Secure communications are accomplished via Secure Sockets Layer (SSL).

An Informant 34 can also send a MessageML document 10 as a SMTP email attachment to a specific SmartMessage account on a mail server hosted by a MessageML Service Provider 18. The Service Provider 18 will process the attached MessageML document and route it to the specified recipients 36. Secure communications are accomplished via S/MIME.

The MessageML architecture 32 handles secure messaging in three ways. The Informant 34 via the SmartMessage Transfer Agent 40 and MessageML Service Provider 18 must support the following authentication and security methods: Informant Stylesheet Authentication, Secure Sockets Layer (SSL), and S/MIME.

With Informant Stylesheet Authentication, the SmartMessage references a specific Informant Stylesheet 14, which contains a list of valid transport sources from which a SmartMessage 10 can be sent. An Informant 34 may have many Informant Stylesheets 14 with different combinations of valid transport sources to be used with different classes of SmartMessages 10.

Using Secure Sockets Layer encryption, SmartMessages can be sent via HTTP to guarantee secure transmission. When posting data to a web page the URL should use SSL. For example, https://... would be used instead of http://...

SmartMessages 10 can also be sent as attachments to SMTP mail, using S/MIME to add cryptographic security services to mail that is sent, and to interpret cryptographic security services in mail that is received. S/MIME needs to be enabled by the sender and supported by the receiver of the mail transaction.

In addition to maintaining versioned XML schemas of standard MessageML documents 10, MessageML.org also manages specific categories of "standard" SmartMessages 10. These include areas such as receipts, query operations, configuration, web messages, and public communications.

Since MessageML.org is the single Informant 34 for these types of standard messages, these standard SmartMessages 10 always refer to MessageML.org's Informant Stylesheet 14 (the *informant-stylesheet-class* and *informant-stylesheet-version* attributes). The following lists the Informant Stylesheet 14 for MessageML.org.

```
<smInformantStylesheet
    xmlns="x-
schema:http://smartmessage.messageml.org/schemas/stylesheets/informant/v1-1.xdr"
    informant-name="MessageML.org"
    informant-stylesheet-
class="http://smartmessage.messageml.org/stylesheets/informant/"
    informant-stylesheet-version="v1-0.xml"
    informant-description="MessageML.org"
    logo-url="http://smartmessage.messageml.org/informants/logo.jpg"
    home-url="http://smartmessage.messageml.org"
```

```
        informant-category="miscellaneous">
      <valid-transport-source transport-protocol="smtp" transport-source="*"/>
      <valid-transport-source transport-protocol="http" transport-source="*"/>
    </smInformantStylesheet>
    <smInformantStylesheet
        xmlns="x-
    schema:http://smartmessage.messageml.org/schemas/stylesheets/informant/v1-1.xdr"
        informant-name="MessageML.org"
        informant-stylesheet-
    class="http://smartmessage.messageml.org/stylesheets/informant/"
        informant-stylesheet-version="v1-0.xml"
        informant-description="MessageML.org"
        logo-url="http://smartmessage.messageml.org/informants/logo.jpg"
        home-url="http://smartmessage.messageml.org"
        informant-category="miscellaneous">
      <valid-transport-source transport-protocol="smtp" transport-source="*"/>
      <valid-transport-source transport-protocol="http" transport-source="*"/>
    </smInformantStylesheet>
    <smInformantStylesheet
        xmlns="x-
    schema:http://smartmessage.messageml.org/schemas/stylesheets/informant/v1-1.xdr"
        informant-name="MessageML.org"
        informant-stylesheet-
    class="http://smartmessage.messageml.org/stylesheets/informant/"
        informant-stylesheet-version="v1-0.xml"
        informant-description="MessageML.org"
        logo-url="http://smartmessage.messageml.org/informants/logo.jpg"
        home-url="http://smartmessage.messageml.org"
        informant-category="miscellaneous">
      <valid-transport-source transport-protocol="smtp" transport-source="*"/>
```

<valid-transport-source transport-protocol="http" transport-source="*"/>
</smInformantStylesheet>

Specifications for these additional standard SmartMessage types 10 can be found at www.MessageML.org. The publishing location and naming convention is as follows.

[informant website url]/stylesheets/[activity]/v[major version]-[minor version].xml

where:

[informant website url] is the URL of the Internet host
[activity] is the activity class that the stylesheet defines
[major version] number for major version number
[minor version] number for minor version number

example: http://www.futureairlines.com/stylesheets/travel-itinerary/v1-0.xml

In this scenario the smartmessage-stylesheet-class would be http://www.futureairlines.com/stylesheets/travel-itinerary/ and the smartmessage-stylesheet-version would be v1-0.xml.

Informant Stylesheets 14 follow a similar publishing location and name convention.

[informant website url]/stylesheets/informant/v[major version]-[minor version].xml

where:

[informant website url] is the URL of the Internet host
[major version] number for major version number
[minor version] number for minor version number

example: http://www.futureairlines.com/stylesheets/informant/v1-0.xml

In this scenario the informant-stylesheet-class would be http://www.futureairlines.com/stylesheets/informant/ and the informant-stylesheet-version would be v1-0.xml.

As shown in Figure 10, a Receipt 48 provides a status update as to the progress of a SmartMessage. The MessageML design utilizes a SmartMessage Stylesheet for receipts 16' in a similar fashion as that shown in figure 4 and described above. The receipt 48 is returned to the Informant 34 through the SmartMessage Transfer Agent 40. The MessageML specification supports three types of receipts 48: received, processed, and delivery status. These receipt types are requested to be sent by creating the appropriate entry or entries under the receipt-request element in the original SmartMessage.

Since receipts 48 are a standard SmartMessage type, their associated Informant Stylesheet 14' and SmartMessage Stylesheet 16' reside on the MessageML Forum website at www.messagemLorg. This ensures that receipts 48 are transacted in a standard way across all implementations.

The SmartMessage 10 enters the received state when it has been received by a MessageML Service Provider (MMLSP) 18. At this point all destination SmartMessage accounts are validated against the MMLSP's account database. Destination SmartMessage accounts not matching the Service Providers domain name are skipped and not included in receipt processing. The following table describes the valid receipt acknowledgements for a received receipt.

Type	Description
ack	The destination SmartMessage was successfully received by the MMLSP for the specified SmartMessage account.
nak	The destination SmartMessage was *not* successfully received by the MMLSP for the specified SmartMessage account. The error-code and error-description attributes and the extended-info element will contain more information.

The SmartMessage 10 enters the processed state when it has been successfully received and is ready to be processed by the MMLSP 18 who performs all the validity

checks and routing rules. The following table describes the valid receipt acknowledgements for a processed receipt.

Type	Description
ack	The destination SmartMessage was successfully processed by the MMLSP for the specified SmartMessage account.
nak	The destination SmartMessage was *not* successfully processed by the MMLSP for the specified SmartMessage account. The error-info and the extended-info elements will contain more information.

The SmartMessage 10 enters the delivery status state when it has been successfully processed and is ready to be delivered. A SmartMessage 10 that cannot be delivered initially enters a message retry loop and continues to be resent until the retry interval or retry time period has expired. The following table describes the valid receipt acknowledgements for a delivery-status receipt.

Type	Description
ack	The destination SmartMessage was successfully delivered by the MMLSP for the specified SmartMessage account.
nak	The destination SmartMessage was *not* successfully delivered by the MMLSP for the specified SmartMessage account. The error-info and the extended-info elements will contain more information.
Retry	The SmartMessage failed to be delivered and the MMLSP is resending the message and will continue to retry sending the message until the retry interval or retry time has expired.

Receipts 48 are delivered at a granular level. They are not grouped and relate to a single destination address receipt request.

The process flow using receipts is shown in figure 11, which assumes that all of the receipt types have been specified. First, an Informant creates a SmartMessage 10, specifying the desired receipt-requests, and passes it to the SmartMessage Application Interface (SMAI) 38. Then, the SmartMessage Transfer Agent (SMTA) 40 queries the SmartMessage 10 and identifies all destination domain names to where the message will be sent. Next, the SMTA 40 makes a copy of the SmartMessage 10' for each destination host and submits the message using the protocol or protocols specified in the

SmartMessage 10. The SMTA 40 handles any failed SMTP or HTTP attempts to reach the "to-address" destinations. The MessageML Service Providers 18 receives the message and validates that all the "to-address" accounts are valid to the MessageML Service Provider 18. Received receipts are delivered to the Informant's SMTA 40 who processes them accordingly. If the SmartMessage 10 is received correctly, it is then processed. Processed receipts are sent to the Informant's SMTA 40 who processes them accordingly. Next, the message is delivered based on the SmartMessage user's predefined endpoint routing rules. Delivered receipts are sent to the Informant's SMTA 40 who processes them accordingly. A per user receipt item will be included for each endpoint the message was delivered.

The following tables disclose the elements and attributes for the received event payload definition, the processed event payload definition and the delivery status event payload definition, respectively.

Received Event Payload Definition

Element	Opt	Multi	Attributes	Sample Value (underline=default)
Receipt				
			receipt-event	received
			receipt-type	ack, nak
			receipt-date	ISO 8601 format (ex: 2000-03-17T15:10:33)
			smartmessage-id	(smartmessage-id from the original SmartMessage)
			to-address	user@domainname (from original SmartMessage)
error-info	X			
			error-class	http, https, smtp, smime, platform-specific
			error-code	(see Error Code section)
			error-description	(see Error Code section)
extended-info	X			

Processed Event Payload Definition

Element	Opt	Multi	Attributes	Sample Value (underline=default)
Receipt	X			
			receipt-event	processed
			receipt-type	ack, nak
			receipt-date	ISO 8601 format (ex: 2000-03-17T15:10:33)
			SmartMessage-id	(smartmessage-id from the original SmartMessage)
			to-address	user@domainname (from original SmartMessage)
error-info	X			
			error-class	http, https, smtp, smime, platform-specific
			error-code	(see Error Code section)
			error-description	(see Error Code section)
extended-info	X			

Delivery Status Event Payload Definition

Element	Opt	Multi	Attributes	Sample Value (underline=default)
receipt	X			
			receipt-event	Received
			receipt-type	ack, nak
			receipt-date	ISO 8601 format (ex: 2000-03-17T15:10:33)
			SmartMessage-id	(smartmessage-id from the original SmartMessage)
			to-address	user@domainname (from original SmartMessage)
			endpoint-type	browser, html-email, text-email, tiny-mail, fax, voice-phone, instant-message
			endpoint-address	endpoint specific (ex: fax number, email address, phone number, etc.
			last-attempt-date	(last delivery attempt) ISO 8601 format.
			will-retry-attempt	(number of retry attempt remaining)
			will-retry-until	(when retry attempts will stop) ISO 8601 format
			next-retry-attempt	(when the next retry attempt will occur)

	[redacted]	[redacted]	ISO 8[illegible] format
error-info	X		
		error-class	http, h[illegible], smtp, smime, platform-specif[illegible]
		error-code	(see E[illegible]ode section)
		error-description	(see E[illegible]ode section)
extended-info	X		

Further, error codes, as desc[illegible]ed in the following table. [illegible]ntained the error-info element of the received, proces[illegible]d and delivery status ever[illegible]oads.

Attribute	Values	Description
error-class		
	http	HTTP/1.1 Status Code. [illegible]ror codes used are defined by W3C (http://www.w3.org/Pro[illegible]/rfc2616/rfc2616-sec10.html#sec10)
	https	same as above
	smtp	SMTP Reply Code. The e[illegible]r codes users are defined by RFC 1893 (http://www.ietf.org/rfc/r[illegible]1893.txt?number=1893) and RFC 821, section 4.[illegible] (http://ietf.org/rfc/rfc082[illegible]t?number=821)
	smime	S/MIME error code
	platform-specific	This class is designated [illegible]mplementers of the SmartMessage specificat[illegible]. Its can be used to denote errors specific to a platform's implementation.
error-code	0=success, non-zero value based on error-class.	Error code associated with the error-class
error-description		(textual description of error)

An example of a Receipt SmartMessage Stylesheet for Receipts is as follows:

SmartMessageStylesheet:

```
xmlns="x-schema:http://sm.smartmessage.org/schemas/stylesheets/smartmessage/v1-1.xdr"
smartmessage-stylesheet-class="http://sm.smartmessage.org/stylesheets/receipts/"
smartmessage-stylesheet-version="v1-0.xml">
```

```
<activity-class
    activity-name="Receipts"
    activity-duration="ongoing"
    event-frequency="once">

    <activity-xsl-default>
        <xsl_default:stylesheet xmlns="http://www.w3.org/TR/REC-html40"
                xmlns:xsl_default="http://www.w3.org/TR/WD-xsl">
            <xsl_default:template match="/"/>
        </xsl_default:stylesheet>
    </activity-xsl-default>

    <event-class
        event-name="DeliveryStatus"
        purpose="information"
        sensitivity="normal"
        reach="individual"
        immediacy="days"
        frequency="daily"
        category="system">

        <event-payload-schema>
            <Schema name="deliverystatusreceipt.xdr"
                xmlns="urn:schemas-microsoft-com:xml-data"
                xmlns:dt="urn:schemas-microsoft-com:datatypes">
                <ElementType name="receipt" content="eltOnly" order="seq">
                    <AttributeType name="receipt-event" dt:type="enumeration"
                            dt:values="delivery-status" default="delivery-status"/>
                    <AttributeType name="receipt-type" dt:type="enumeration"
                            dt:values="ack nak retry" default="nak"/>
                    <AttributeType name="receipt-date" dt:type="dateTime"
                            required="yes"/>
                    <AttributeType name="smartmessage-id" dt:type="string"
                            required="yes"/>
                    <AttributeType name="to-address" dt:type="string"
required="yes"/>
                    <AttributeType name="endpoint-type" dt:type="enumeration"
                            dt:values="browser html-email text-email tiny-email fax
                            voice-phone instant-message" required="yes"/>
                    <AttributeType name="endpoint-address" dt:type="string"
                            required="yes"/>
                    <AttributeType name="last-attempt-date" dt:type="dateTime"
                            required="yes"/>
                    <AttributeType name="will-retry-attempt" dt:type="int"
```

```
                    required="yes"/>
                <AttributeType name="will-retry-until" dt:type="dateTime"
                    required="yes"/>
                <AttributeType name="next-retry-attempt" dt:type="dateTime"
                    required="yes"/>
                <attribute type="receipt-event"/>
                <attribute type="receipt-type"/>
                <attribute type="receipt-date"/>
                <attribute type="smartmessage-id"/>
                <attribute type="to-address"/>
                <attribute type="endpoint-type"/>
                <attribute type="endpoint-address"/>
                <attribute type="last-attempt-date"/>
                <attribute type="will-retry-attempt"/>
                <attribute type="will-retry-until"/>
                <attribute type="next-retry-attempt"/>
                <element type="error-info" minOccurs="0" maxOccurs="1"/>
                <element type="extended-info" minOccurs="0" maxOccurs="1"/>
            </ElementType>
            <ElementType name="error-info" content="eltOnly" order="seq">
                <AttributeType name="error-class" dt:type="enumeration"
                    dt:values="http https smtp smime platform-specific"
                    default="smtp" required="yes"/>
                <AttributeType name="error-code" dt:type="string"
required="yes"/>
                <AttributeType name="error-description" dt:type="string"
                    required="yes"/>
                <attribute type="error-class"/>
                <attribute type="error-code"/>
                <attribute type="error-description"/>
            </ElementType>
            <ElementType name="extended-info" content="eltOnly"
order="seq"/>
        </Schema>
    </event-payload-schema>
    <event-xsl-default>
        <xsl_default:stylesheet xmlns="http://www.w3.org/TR/REC-html40"
            xmlns:xsl_default="http://www.w3.org/TR/WD-xsl">
            <xsl_default:template match="/">
                <xsl_default:apply-templates select="receipt"/>
            </xsl_default:template>
            <xsl_default:template match="receipt">
Receipt Item
    receipt-event:      <xsl_default:value-of select="@receipt-event"/>
    receipt-type:       <xsl_default:value-of select="@receipt-type"/>
```

```
receipt-date:          <xsl_default:value-of select="@receipt-date"/>
smartmessage-id:       <xsl_default:value-of select="@smartmessage-id"/>
to-address:            <xsl_default:value-of select="@to-address"/>
endpoint-type:         <xsl_default:value-of select="@endpoint-type"/>
endpoint-address:      <xsl_default:value-of select="@endpoint-address"/>
last-attempt-date:     <xsl_default:value-of select="@last-attempt-date"/>
will-retry-attempt: <xsl_default:value-of select="@will-retry-attempt"/>
will-retry-until:      <xsl_default:value-of select="@will-retry-until"/>
next-retry-attempt:    <xsl_default:value-of select="@next-retry-attempt"/>
error-class:        <xsl_default:value-of select="error-info/@error-class"/>
error-code:            <xsl_default:value-of select="error-info/@error-code"/>
error-description:     <xsl_default:value-of select="error-info/@error-description"/>

Receipt Specific Data:
  <xsl_default:value-of select="extended-info"/>
            </xsl_default:template>
          </xsl_default:stylesheet>
        </event-xsl-default>
      </event-class>

      <event-class
          event-name="Received"
          purpose="information"
          sensitivity="normal"
          reach="individual"
          immediacy="days"
          frequency="once"
          category="miscellaneous">

          <event-payload-schema>
              <Schema name="receivedreceipt.xdr"
                  xmlns="urn:schemas-microsoft-com:xml-data"
                  xmlns:dt="urn:schemas-microsoft-com:datatypes">
                  <ElementType name="receipt" content="eltOnly" order="seq">
                      <AttributeType name="receipt-event" dt:type="enumeration"
                              dt:values="received" default="received"/>
                      <AttributeType name="receipt-type" dt:type="enumeration"
                              dt:values="ack nak" default="nak"/>
                      <AttributeType name="receipt-date" dt:type="dateTime"
                              required="yes"/>
                      <AttributeType name="smartmessage-id" dt:type="string"
                              required="yes"/>
                      <AttributeType name="to-address" dt:type="string"
required="yes"/>
```

```
                    <AttributeType name="error-code" dt:type="string"
required="no"/>
                    <AttributeType name="error-description" dt:type="string"
                        required="no"/>
                    <attribute type="receipt-event"/>
                    <attribute type="receipt-type"/>
                    <attribute type="receipt-date"/>
                    <attribute type="smartmessage-id"/>
                    <attribute type="to-address"/>
                    <element type="error-info" minOccurs="0" maxOccurs="1"/>
                    <element type="extended-info" minOccurs="0" maxOccurs="1"/>
                </ElementType>
                <ElementType name="error-info" content="eltOnly" order="seq">
                    <AttributeType name="error-class" dt:type="enumeration"
                        dt:values="http https smtp smime platform-specific"
                        default="smtp" required="yes"/>
                    <AttributeType name="error-code" dt:type="string"
required="yes"/>
                    <AttributeType name="error-description" dt:type="string"
                        required="yes"/>
                    <attribute type="error-class"/>
                    <attribute type="error-code"/>
                    <attribute type="error-description"/>
                </ElementType>
                <ElementType name="extended-info" content="eltOnly"
order="seq"/>
            </Schema>
        </event-payload-schema>
        <event-xsl-default>
            <xsl_default:stylesheet xmlns="http://www.w3.org/TR/REC-html40"
                xmlns:xsl_default="http://www.w3.org/TR/WD-xsl">
                <xsl_default:template match="/">
                    <xsl_default:apply-templates select="receipt"/>
                </xsl_default:template>
                <xsl_default:template match="receipt">
Receipt Item
    receipt-event:          <xsl_default:value-of select="@receipt-event"/>
    receipt-type:           <xsl_default:value-of select="@receipt-type"/>
    receipt-date:           <xsl_default:value-of select="@receipt-date"/>
    smartmessage-id:        <xsl_default:value-of select="@smartmessage-id"/>
    to-address:             <xsl_default:value-of select="@to-address"/>
    error-class:          <xsl_default:value-of select="error-info/@error-class"/>
    error-code:             <xsl_default:value-of select="error-info/@error-code"/>
    error-description:      <xsl_default:value-of select="error-info/@error-description"/>
```

```
Receipt Specific Data:
  <xsl_default:value-of select="extended-info"/>
                </xsl_default:template>
              </xsl_default:stylesheet>
            </event-xsl-default>
          </event-class>

          <event-class
              event-name="Processed"
              purpose="information"
              sensitivity="normal"
              reach="individual"
              immediacy="days"
              frequency="once"
              category="miscellaneous">

              <event-payload-schema>
                  <Schema name="processedreceipt.xdr"
                      xmlns="urn:schemas-microsoft-com:xml-data"
                      xmlns:dt="urn:schemas-microsoft-com:datatypes">
                      <ElementType name="receipt" content="eltOnly" order="seq">
                          <AttributeType name="receipt-event" dt:type="enumeration"
                                  dt:values="processed" default="processed"/>
                          <AttributeType name="receipt-type" dt:type="enumeration"
                                  dt:values="ack nak" default="nak"/>
                          <AttributeType name="receipt-date" dt:type="dateTime"
                                  required="yes"/>
                          <AttributeType name="smartmessage-id" dt:type="string"
                                  required="yes"/>
                          <AttributeType name="to-address" dt:type="string"
      required="yes"/>
                          <AttributeType name="error-code" dt:type="string"
      required="no"/>
                          <AttributeType name="error-description" dt:type="string"
                                  required="no"/>
                          <attribute type="receipt-event"/>
                          <attribute type="receipt-type"/>
                          <attribute type="receipt-date"/>
                          <attribute type="smartmessage-id"/>
                          <attribute type="to-address"/>
                          <element type="error-info" minOccurs="0" maxOccurs="1"/>
                          <element type="extended-info" minOccurs="0" maxOccurs="1"/>
                      </ElementType>
                      <ElementType name="error-info" content="eltOnly" order="seq">
                          <AttributeType name="error-class" dt:type="enumeration"
```

```
                    dt:values="http https smtp smime platform-specific"
                    default="smtp" required="yes"/>
                <AttributeType name="error-code" dt:type="string"
required="yes"/>
                <AttributeType name="error-description" dt:type="string"
                    required="yes"/>
                <attribute type="error-class"/>
                <attribute type="error-code"/>
                <attribute type="error-description"/>
              </ElementType>
              <ElementType name="extended-info" content="eltOnly"
order="seq"/>
            </Schema>
          </event-payload-schema>
          <event-xsl-default>
            <xsl_default:stylesheet xmlns="http://www.w3.org/TR/REC-html40"
              xmlns:xsl_default="http://www.w3.org/TR/WD-xsl">
              <xsl_default:template match="/">
                <xsl_default:apply-templates select="receipt"/>
              </xsl_default:template>
              <xsl_default:template match="receipt">
Receipt Item
  receipt-event:      <xsl_default:value-of select="@receipt-event"/>
  receipt-type:       <xsl_default:value-of select="@receipt-type"/>
  receipt-date:       <xsl_default:value-of select="@receipt-date"/>
  smartmessage-id:    <xsl_default:value-of select="@smartmessage-id"/>
  to-address:         <xsl_default:value-of select="@to-address"/>
  error-class:      <xsl_default:value-of select="error-info/@error-class"/>
  error-code:         <xsl_default:value-of select="error-info/@error-code"/>
  error-description:  <xsl_default:value-of select="error-info/@error-description"/>
  Receipt Specific Data:
    <xsl_default:value-of select="extended-info"/>
              </xsl_default:template>
            </xsl_default:stylesheet>
          </event-xsl-default>
        </event-class>
      </activity-class>
    </smSmartMessageStylesheet>
```


Although some of MessageML's initial applications and services are targeted towards providing electronic messaging to Recipient endpoints devices 12, there is opportunity to apply the standard to a multitude of other applications, i.e., two-way

messaging communication between automated processes, transformation to and from other XML formats to provide messaging interoperability with other XML technologies, communication routing applications for field service, etc.

The foregoing detailed description of the invention is intended to be illustrative and not intended to limit the scope of the invention. Changes and modifications are possible with respect to the foregoing description, and it is understood that the invention may be practiced otherwise than that specifically described herein and still be within the scope of the claims.

CLAIMS

1. A method for providing content driven electronic messaging that enables individuals to receive XML electronic messages using a electronic messaging system, comprising the steps of; creating an Informant Stylesheet; creating at least one SmartMessage Stylesheet; creating at least one SmartMessage; sending said at least one SmartMessage to a MessageML Service Provider; receiving said at least one SmartMessage; processing said at least one SmartMessage, and delivering said SmartMessage to at least one endpoint of a user of a said electronic messaging system, based on a definition created by said user.

2. The method for providing content driven electronic messaging in claim 1, wherein said step of creating an Informant Stylesheet comprises defining meta-data about the Informant.

3. The method for providing content driven electronic messaging in claim 2, wherein said meta-data about the Informant comprises locations from where a SmartMessage may originate.

4. The method for providing content driven electronic messaging in claim 2, wherein said step of creating an Informant Stylesheet comprises storing said Informant Stylesheet on the Informant's web server.

5. The method for providing content driven electronic messaging in claim 1, wherein said step of creating at least one SmartMessage Stylesheet comprises defining a message payload and how said message payload is to be displayed at said at least one endpoint.

6. The method for providing content driven electronic messaging in claim 5, wherein said message payload is defined for an event class by defining an XML schema.

7. The method for providing content driven electronic messaging in claim 5, wherein said how said message payload is to be displayed is defined by an XSL document.

8. The method for providing content driven electronic messaging in claim 5, wherein said step of creating at least one SmartMessage Stylesheet comprises storing said at least one SmartMessage Stylesheet on the Informant's web server.

9. The method for providing content driven electronic messaging in claim 1, wherein said at least one SmartMessage comprises said SmartMessage Stylesheet, said Informant Stylesheet, and a message payload.

10. The method for providing content driven electronic messaging in claim 9, wherein said message payload comprises an activity payload and an event payload.

11. The method for providing content driven electronic messaging in claim 10, wherein said event payload pertains to said activity payload.

12. The method for providing content driven electronic messaging in claim 9, wherein said at least one SmartMessage comprises an XML document.

13. The method for providing content driven electronic messaging in claim 1, wherein said step of sending said at least one SmartMessage to a MessageML Service Provider comprises the transmission of a SmartMessage from said Informant to said MessageML Service Provider.

14. The method for providing content driven electronic messaging in claim 1, wherein said step of receiving said at least one SmartMessage comprises said MessageML Service Provider receiving said SmartMessage, processing said at least one SmartMessage based on said Informant Stylesheet and said at least one SmartMessage Stylesheet, and delivering said SmartMessage to at least one endpoint of a user of a said electronic messaging system, based on a definition created by said user.

15. The method for providing content driven electronic messaging in claim 1, wherein said step of processing said at least one SmartMessage comprises said MessageML Service Provider processing based on said Informant Stylesheet and said at least one SmartMessage Stylesheet.

16. The method for providing content driven electronic messaging in claim 1, further comprising the step of signing up to receive said at least one SmartMessage from an Informant.

17. The method for providing content driven electronic messaging in claim 16, wherein said step of signing up to receive said at least one SmartMessage from an Informant comprises deciding where said at least one SmartMessage is to be delivered.

18. The method for providing content driven electronic messaging in claim 17, wherein said deciding where said at least one SmartMessage is to be delivered, comprises deciding said at least one endpoint of said user of said electronic messaging system.

19. The method for providing content driven electronic messaging in claim 18, wherein said at least one endpoint comprises an electronic mail box, a wired or wireless telephone, a facsimile machine, a paging device, a web site or a personal digital assistant.

20. The method for providing content driven electronic messaging in claim 19, wherein said step of delivering said at least one SmartMessage comprises transmitting said at least one SmartMessage to said endpoint.

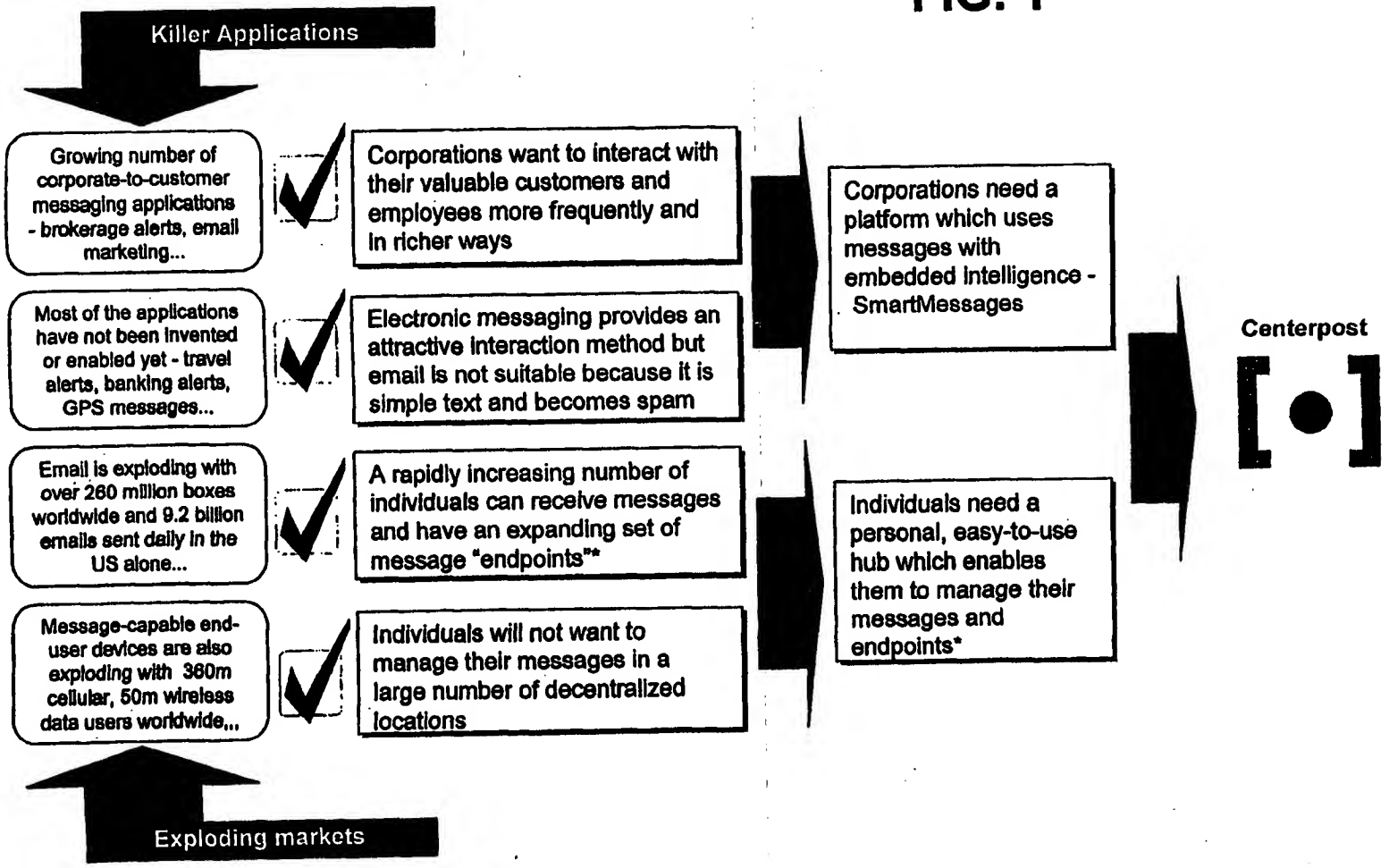
21. The method for providing content driven electronic messaging in claim 1, further comprising the step of creating a SmartMessage Stylesheet for receipts, said receipts are capable of providing a status update as to the progress of the SmartMessage.

22. The method for providing content driven electronic messaging in claim 21, wherein said receipt comprises a receipt type.

23. The method for providing content driven electronic messaging in claim 21, wherein said receipt type comprises received, processed or delivery status.

MARKET NEED FOR CENTERPOST

FIG. 1



*Endpoints are email boxes, digital cellular phones, pagers, instant messages, fax machines, telephones, PDAs, and other such devices

FIG. 2

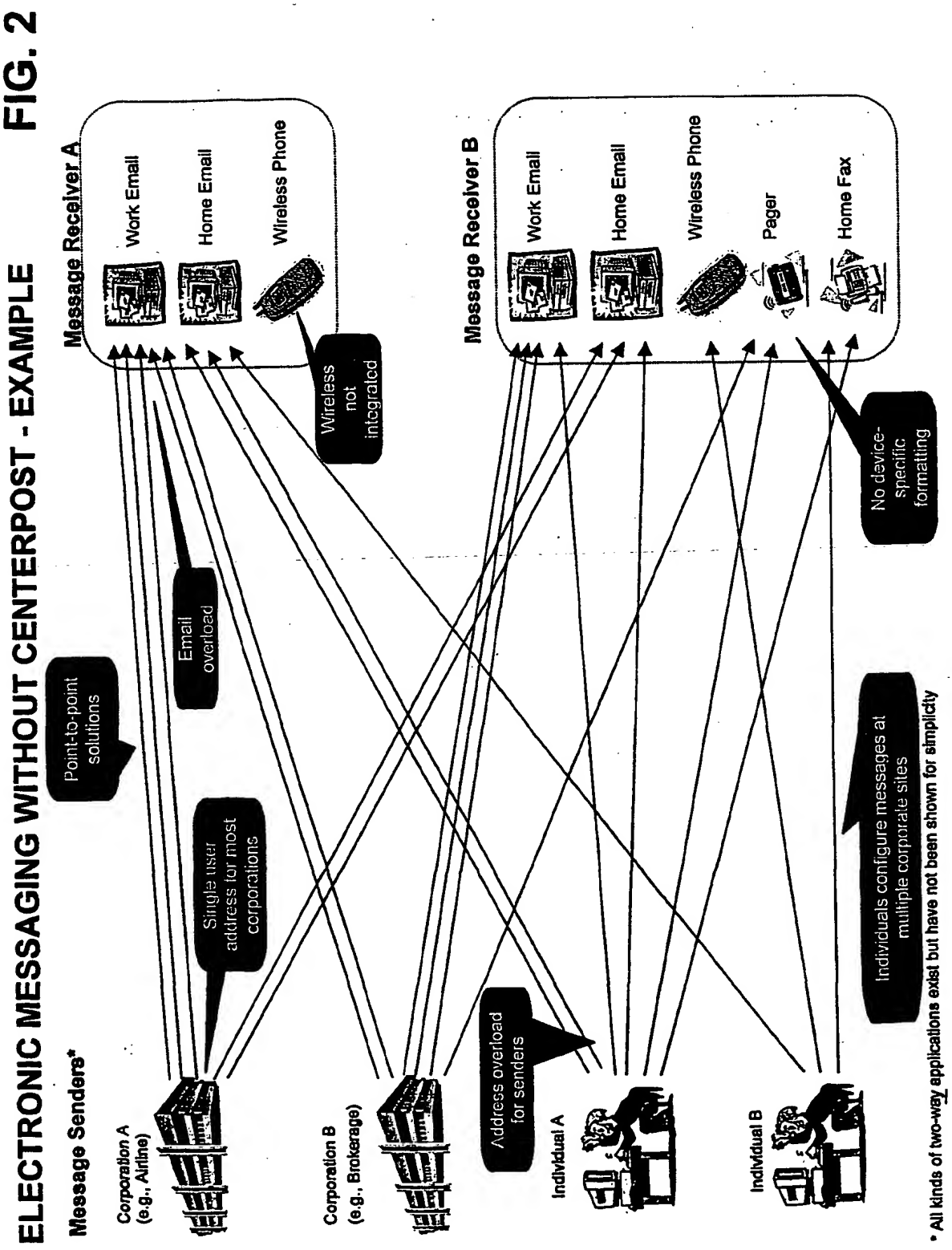
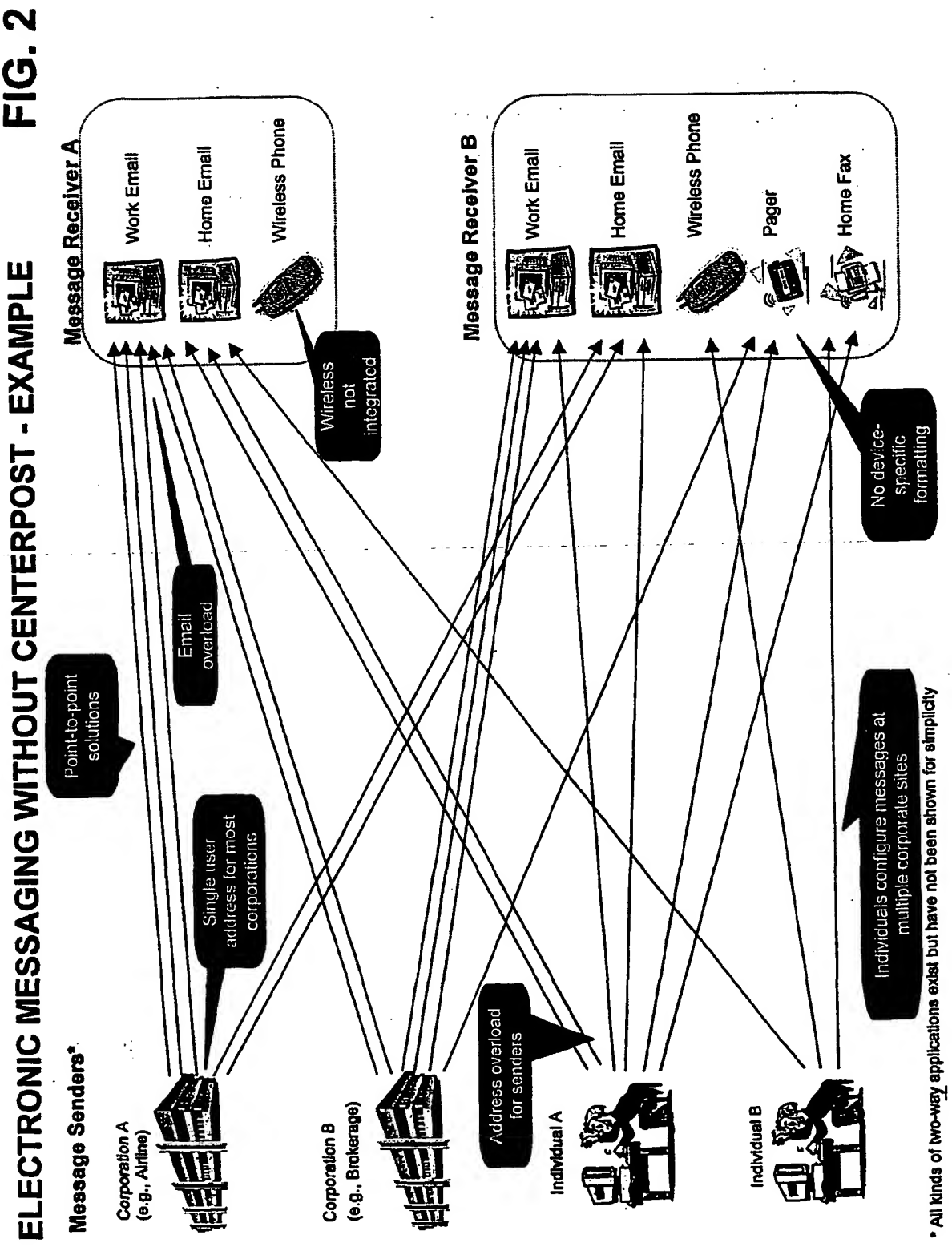

ELECTRONIC MESSAGING WITHOUT CENTERPOST - EXAMPLE
Message Senders*
Corporation A (e.g., Airline)
Corporation B (e.g., Brokerage)
Individual A
Individual B
Point-to-point solutions
Email overload
Single user address for most corporations
Address overload for senders
Individuals configure messages at multiple corporate sites
Message Receiver A
Work Email
Home Email
Wireless Phone
Wireless not integrated
Message Receiver B
Work Email
Home Email
Wireless Phone
Pager
Home Fax
No device-specific formatting
* All kinds of two-way applications exist but have not been shown for simplicity

FIG. 2


ELECTRONIC MESSAGING WITHOUT CENTERPOST - EXAMPLE
Message Senders*
Corporation A (e.g., Airline)
Corporation B (e.g., Brokerage)
Individual A
Individual B
Message Receiver A
Work Email
Home Email
Wireless Phone
Message Receiver B
Work Email
Home Email
Wireless Phone
Pager
Home Fax
Point-to-point solutions
Email overload
Single user address for most corporations
Wireless not integrated
Address overload for senders
Individuals configure messages at multiple corporate sites
No device-specific formatting
* All kinds of two-way applications exist but have not been shown for simplicity

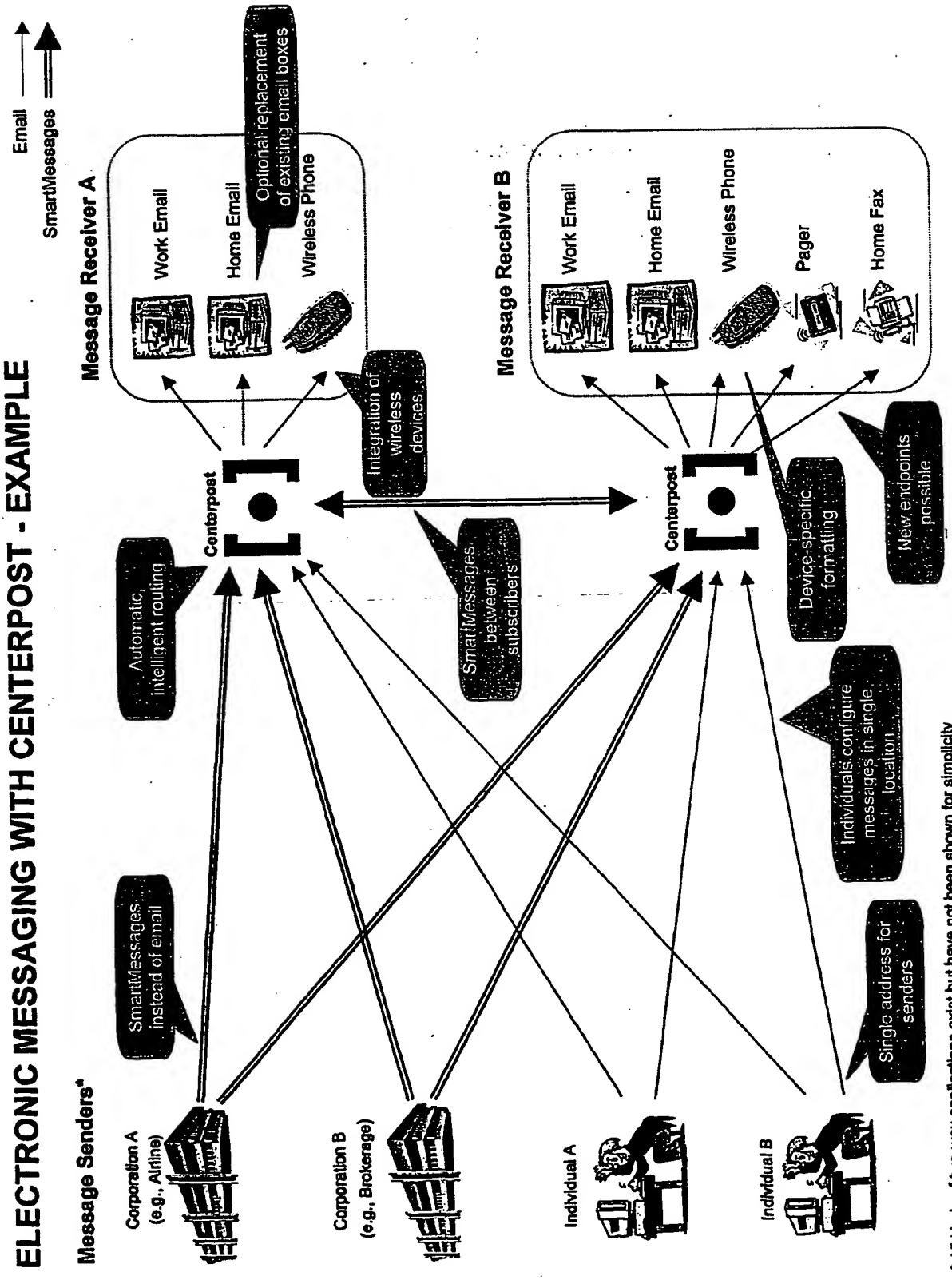
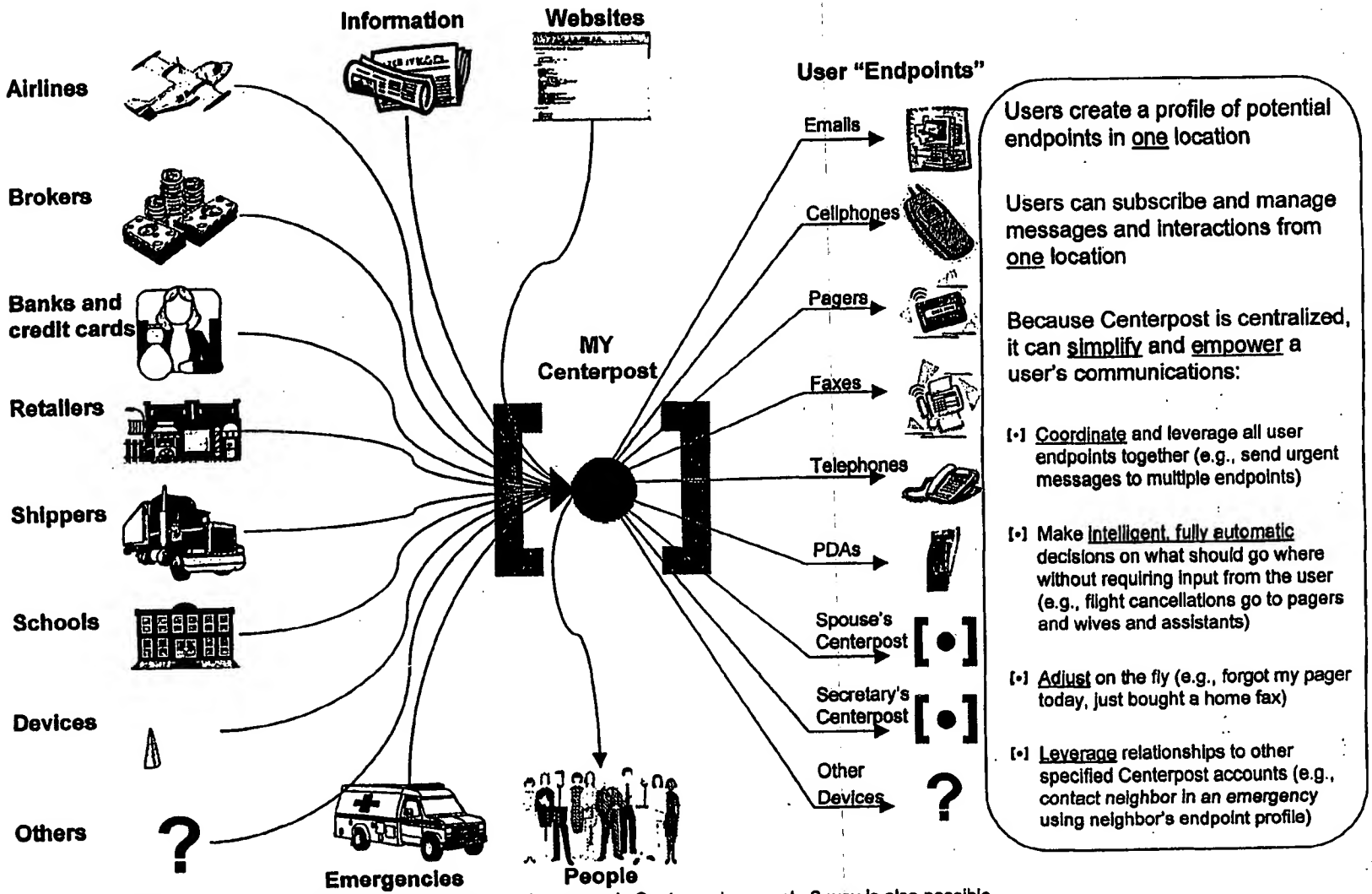

ELECTRONIC MESSAGING WITH CENTERPOST - EXAMPLE
Email
SmartMessages
Message Senders*
Corporation A (e.g., Airline)
Corporation B (e.g., Brokerage)
Individual A
Individual B
SmartMessages instead of email
Automatic, intelligent routing
Centerpost
Integration of wireless devices
SmartMessages between subscribers
Centerpost
Device-specific formatting
New endpoints possible
Individuals configure messages in single location
Single address for senders
Message Receiver A
Work Email
Home Email
Wireless Phone
Optional replacement of existing email boxes
Message Receiver B
Work Email
Home Email
Wireless Phone
Pager
Home Fax
* All kinds of two-way applications exist but have not been shown for simplicity

FIG. 3A

THE POWER OF A CENTRALIZED, USER-CENTRIC HUB FOR CONSUMERS



Note: For simplicity, this diagram reflects one-way interaction into a user's Centerpost account. 2-way is also possible.

FIG. 3B

EXAMPLE CENTERPOST INTERACTIONS FOR AN INDIVIDUAL SUBSCRIBER

	CONTENT	EXAMPLE PROCESSING RULES (for 1 subscriber)	2-WAY POTENTIAL
Airlines 	Booking confirmation	Send to assistant's email; update "folio" on Centerpost	
	Destination weather forecast	Send to home fax on evening prior to trip	
	Flight delay / cancellation	Send to pager and copy assistant's fax; to mobile phone on weekends	Rebook other flight
	Gate assignments	Send to pager; update PalmPilot schedule	
	Destination traffic alert	Send to pager; send to assistant's fax	
	Frequent flyer posting	Consolidate and send weekly summary to home email	
Brokers 	Portfolio news	Send to home email; send urgent taglines to pager	Send details
	Closing prices	Send chart to work email; update "folio" on Centerpost; enable query	
	Stock alerts	Send to home email; send to PCS phone for extreme movements	Execute trade
	Trade alerts	Send to pager; request shares to buy/sell	Execute trade
	Trade expiration	Send to home email	Renew position
Banks and credit cards 	Low balance	Send to home fax	Transfer funds
	Unusual activity	Send to pager; call home phone	Freeze account
	Account postings	Update "folio" on Centerpost;update Quicken register	
Retailers 	Special promotions	Consolidate and send weekly summary on Fridays to home email	Purchase item
	Time sensitive promotions	Send to home email; delete when expired	Purchase item
	Ordered items arrived	Send to home fax	Hold at store
	Ordered items shipped	Send to home fax	
Shippers 	Package signed for	Send to work email "shipments" folder; send to assistant's email	
	Package tracking	Update "folio" on Centerpost; enable query to folio	
Schools 	Emergency contact	Send to all devices; also send to wife's Centerpost account	
	Test scores	Update "folio" on Centerpost	
	School events	Send to home email	
Technicians 	Trouble alert	Follow escalation schedule; update central "folio"	Will respond
	Maintenance notification	Send to work email	
	Scheduling message	Send to pager	
Devices 	Security system alert	Send to pager; ping wife's Centerpost; call neighbor phone	False alarm
	Power outage	Send to pager or cellphone	
	Equipment/network alarms	Follow escalation schedule; update central "folio"	Will respond
	GPS transmission	Update map on work PC	
People 	Email	Route based on various criteria; send subject to pager if urgent	Reply to email
	Address update (from assistant)	Update address book on Palm VII, PCS phone, and home email	
	Schedule change (from assistant)	Update schedule on Palm VII; send to pager if urgent	

FIG. 3C

FIG. 4

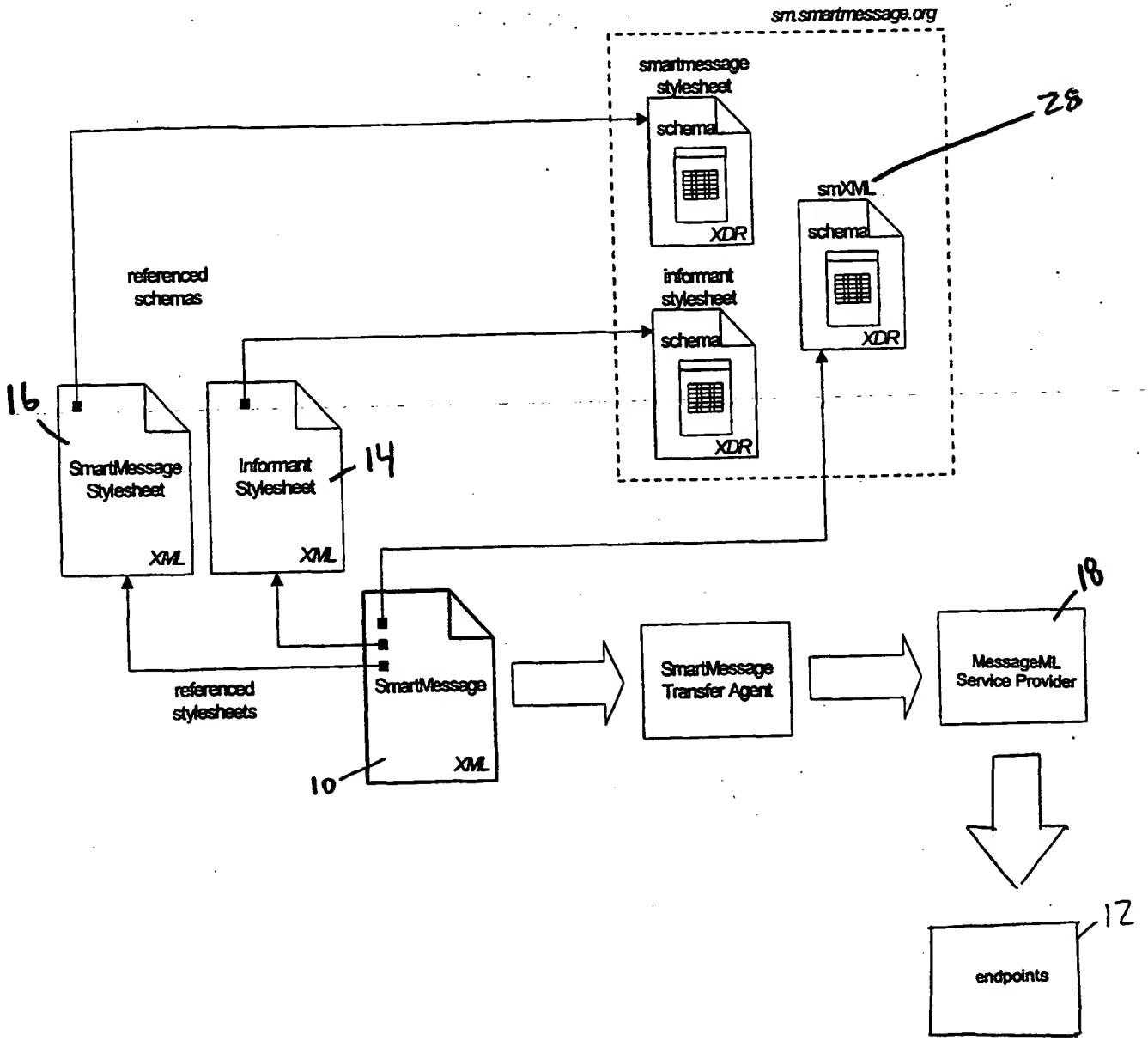

sm.smartmessage.org
smartmessage stylesheet
schema
XDR
smXML
schema
XDR
28
informant stylesheet
schema
XDR
referenced schemas
16
SmartMessage Stylesheet
XML
Informant Stylesheet
XML
14
referenced stylesheets
SmartMessage
XML
10
SmartMessage Transfer Agent
MessageML Service Provider
18
endpoints
12

FIG. 5

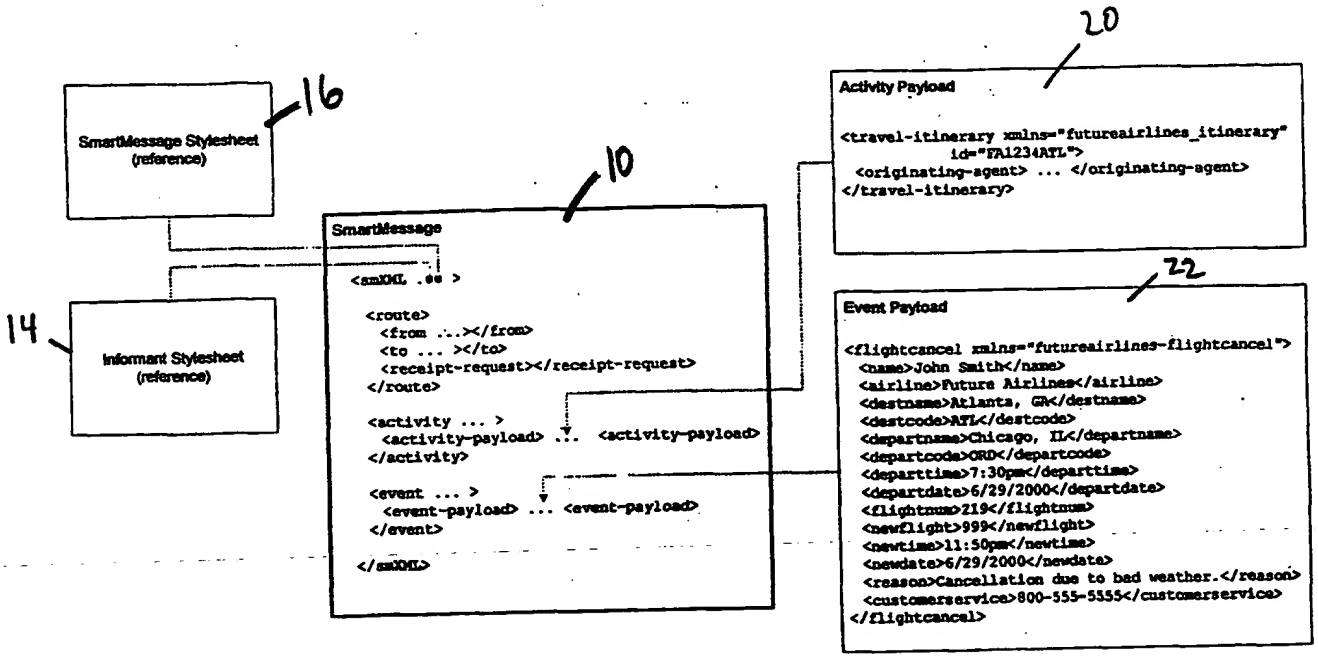

20
Activity Payload
<travel-itinerary xmlns="futureairlines_itinerary"
id="FA1234ATL">
<originating-agent> ... </originating-agent>
</travel-itinerary>
16
SmartMessage Stylesheet
(reference)
10
SmartMessage
<smXML .oo >
<route>
<from ...></from>
<to ... ></to>
<receipt-request></receipt-request>
</route>
<activity ... >
<activity-payload> ... <activity-payload>
</activity>
<event ... >
<event-payload> ... <event-payload>
</event>
</smXML>
14
Informant Stylesheet
(reference)
22
Event Payload
<flightcancel xmlns="futureairlines-flightcancel">
<name>John Smith</name>
<airline>Future Airlines</airline>
<destname>Atlanta, GA</destname>
<destcode>ATL</destcode>
<departname>Chicago, IL</departname>
<departcode>ORD</departcode>
<departtime>7:30pm</departtime>
<departdate>6/29/2000</departdate>
<flightnum>219</flightnum>
<newflight>999</newflight>
<newtime>11:50pm</newtime>
<newdate>6/29/2000</newdate>
<reason>Cancellation due to bad weather.</reason>
<customerservice>800-555-5555</customerservice>
</flightcancel>

FIG. 6

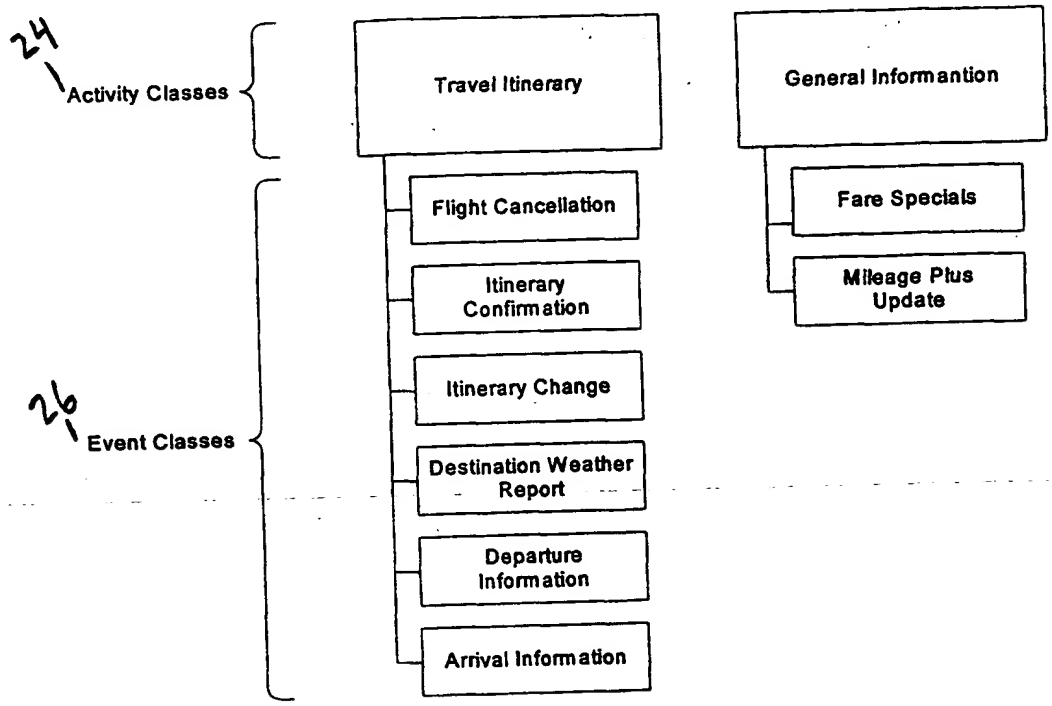

24
Activity Classes
26
Event Classes
Travel Itinerary
General Information
Flight Cancellation
Itinerary Confirmation
Itinerary Change
Destination Weather Report
Departure Information
Arrival Information
Fare Specials
Mileage Plus Update

FIG. 7

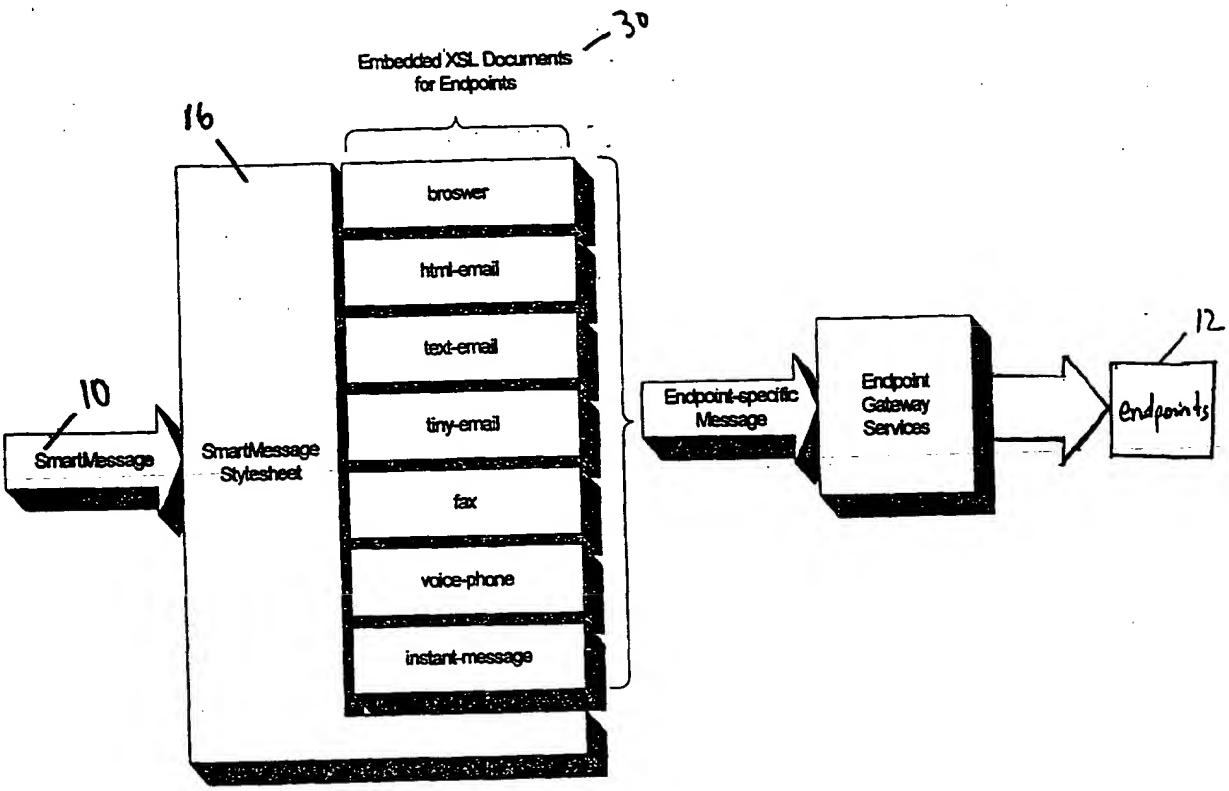

Embedded XSL Documents for Endpoints
30
16
broswer
html-email
text-email
tiny-email
fax
voice-phone
instant-message
10
SmartMessage
SmartMessage Stylesheet
Endpoint-specific Message
Endpoint Gateway Services
12
endpoints

FIG. 8

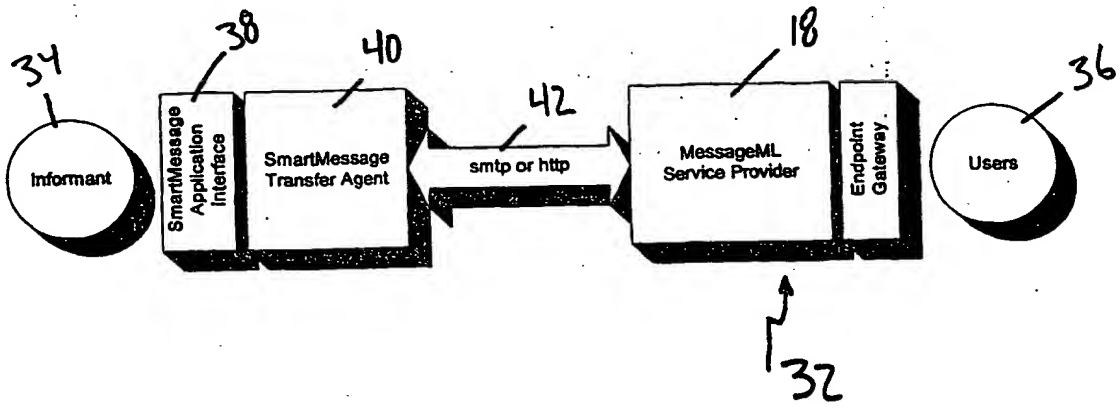

34
38
40
18
42
36
Informant
SmartMessage Application Interface
SmartMessage Transfer Agent
smtp or http
MessageML Service Provider
Endpoint Gateway
Users
32

FIG. 9

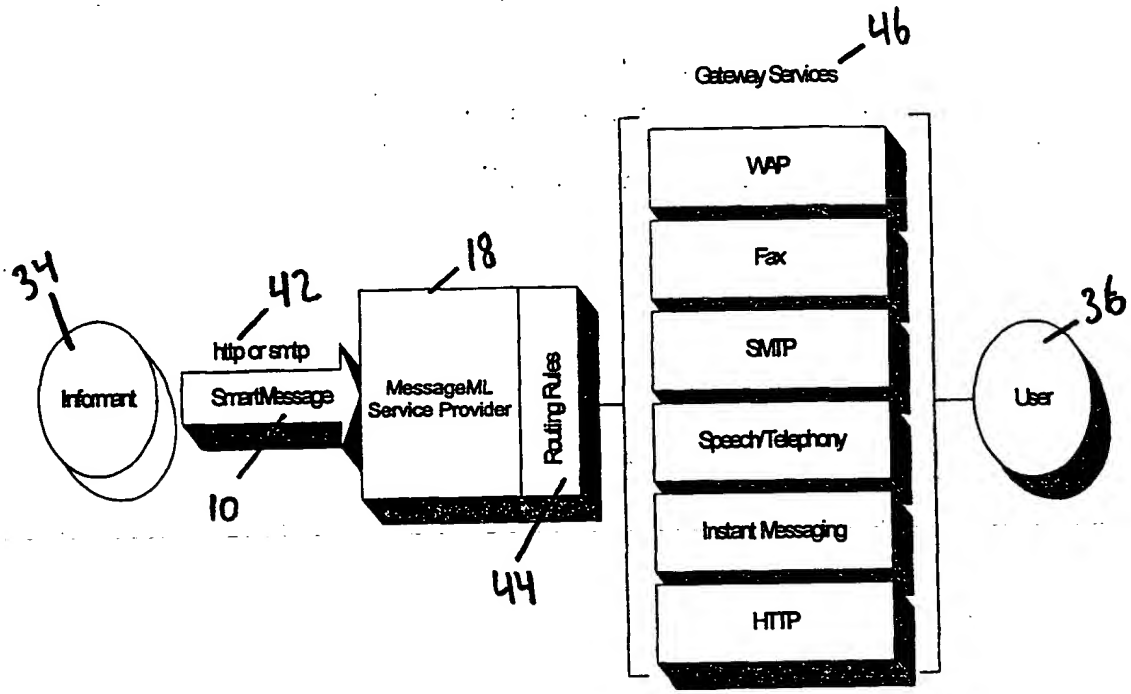

46
Gateway Services
WAP
Fax
SMTP
Speech/Telephony
Instant Messaging
HTTP
34
Informant
42
http or smtp
SmartMessage
10
18
MessageML
Service Provider
Routing Rules
44
36
User

FIG. 10

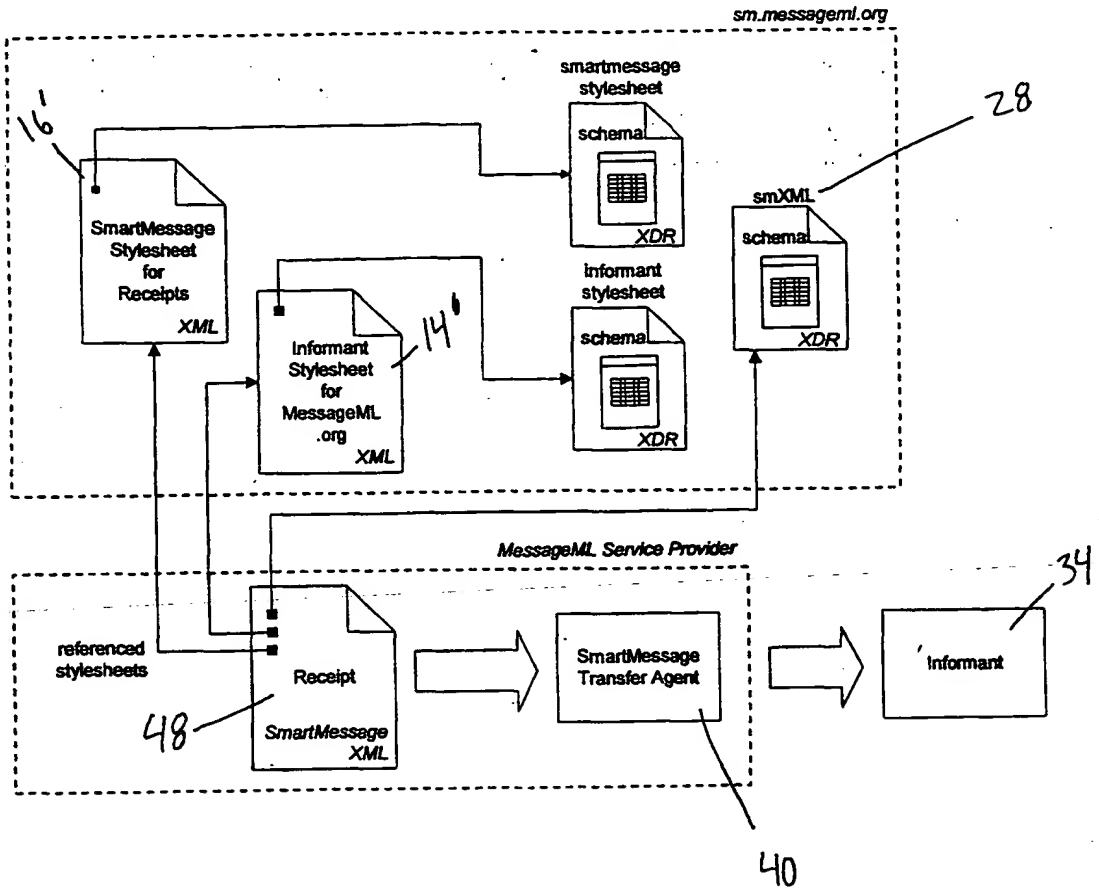
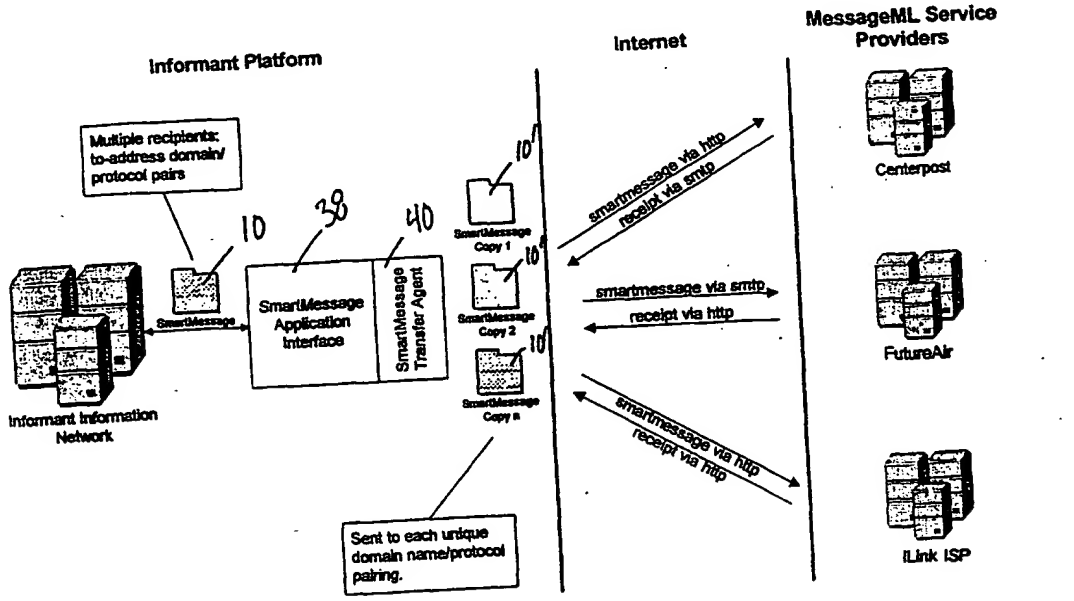

sm.messageml.org
smartmessage stylesheet
schema
XDR
16'
28
smXML
schema
XDR
SmartMessage Stylesheet for Receipts
XML
informant stylesheet
schema
XDR
Informant Stylesheet for MessageML .org
XML
14'
MessageML Service Provider
34
referenced stylesheets
Receipt
SmartMessage
XML
48
SmartMessage Transfer Agent
Informant
40

FIG. 11


18
Informant Platform
Internet
MessageML Service Providers
Multiple recipients: to-address domain/ protocol pairs
10
38
40
SmartMessage Application Interface
SmartMessage Transfer Agent
SmartMessage
Informant Information Network
SmartMessage Copy 1
SmartMessage Copy 2
SmartMessage Copy n
smartmessage via http
receipt via smtp
smartmessage via smtp
receipt via http
smartmessage via http
receipt via http
Centerpost
FutureAir
iLink ISP
Sent to each unique domain name/protocol pairing.